Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

1984

CHART AND SKETCH : an interactive computer graphics program.

Tschudy, James J., II

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/19187

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

T218332

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

CHART AND SKETCH
AN INTERACTIVE COMPUTER GRAPHICS PROGRAM

by

James J. Tschudy II

March 1984

Thesis Advisor: G. R. Porter

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: CHART AND SKETCH: An Interactive Computer Graphics Program	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis; March 1984
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): James J. Tschudy II	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	12. REPORT DATE: March 1984
	13. NUMBER OF PAGES: 146
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Graphics
Interactive
Program

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Chart and Sketch is a highly interactive, color, computer graphics program. It is a Fortran 77 implementation of the Precision Visual's DI-3000 software package and the RAMTEK RM-9460 Graphic Display System. DI-3000 is an integrated system of device independent, graphics software tools that may be called as subroutines in fortran programs.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

The Sketch option provides for the interactive creation of multi-colored, empty and color-filled graphic images including circules, diamonds, squares, polygons, lines, arcs, sectors and text. Created images may be selectively copied, moved, erased, displayed, and written to or read from selected graphics files.

The Chart option allows the creation and display of charts from a file database. Sketch may be used to add graphic images to the displayed chart, which may in turn be written to a graphics file for later recall and display.

The program is written in structured FORTRAN 77, with numerous comments so that it can be read, understood, modified, and expanded with a minimum of effort.

Approved for public release; distribution unlimited.

CHART AND SKETCH: An Interactive Computer Graphics Program

by

James J. Tschudy II
Captain, United States Air Force
B.S., Utah State University, 1978

Submitted in partial fullfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(Command, Control, and Communications)

from the

NAVAL POSTGRADUATE SCHOOL
March 1984

ABSTRACT

Chart and Sketch is a highly interactive, color, computer graphics program. It is implemeted in Fortran 77 and interfaced to the Precision Visual's DI-3000 software package which drives the RAMTEK RM-9460 Graphic Display System. DI-3000 is an integrated system of device independent, graphics software tools that may be called as subroutines in fortran programs.

The Sketch option provides for the interactive creation of multi-colored, empty and color-filled graphic objects such as circles, diamonds, squares, polygons, lines, arcs, sectors and text. Created objects may be selectively scaled, translated, copied, erased, displayed, and written to or read from selected graphics files. Created objects may be combined to form graphics segments. The program provides for within-segment modelling, zooming, and panning.

The Chart option allows the creation and display of charts from a chart database. Sketch may be used to add graphic objects to the displayed chart, which may in turn be written to a graphics file for later retrieval and display.

The program is written in structured FORTRAN 77, internally documented, so that it can be read, understood, modified, and expanded with a minimum of effort.

TABLE OF CONTENTS

I. BACKGROUND 8

II. PROGRAM DESCRIPTION 10

A. PROGRAM DESIGN CONCEPTS 12

1. Simplicity 12

2. Minimization of Memorization 12

3. Consistency 13

4. Feedback 13

5. Error Accommodation 14

6. Response Time 15

B. PROGRAM ALGORITHM 17

1. Main Program Algorithm 17

a. Chart 20

b. Sketch 21

c. File 21

d. Retrieve 21

e. Clear 21

f. View 21

g. Move 21

h. Zoom 22

i. Quit 22

2. Subroutine Algorithms 22

a. Subroutine Initz 22

b. Subroutine Menu 24
c. Subroutine Draw 25
d. Subroutine Filclr 30
e. Subroutine Text 30
f. Subroutine Sketch 30
g. Subroutine Clear 32
h. Subroutine File_sec 33
i. Subroutine File_data 33
j. Subroutine Retrieve_sec 34
k. Subroutine Retrieve_data 35
l. Procedure Points 36

III. PROGRAM IMPLEMENTATION 39

A. USING MENUS TO INVOKE FUNCTIONS 39
B. MAKING SEGMENTS VISIBLE AND INVISIBLE 44
C. TRANSLATING AND SCALING SEGMENTS 47
D. ZOOMING AND PANNING 50
E. FILING GRAPHICS 53

IV. USE OF CHART AND SKETCH. 65

A. IN THE COMMAND AND CONTROL ENVIRONMENT 65
B. AS A GENERAL APPLICATION GRAPHICS TOOL 69

V. RECOMMENDED ENHANCEMENTS 70

A. WORLDWIDE CHART DATABASE ACCESS 70
B. MULTIPLE SEGMENT OBJECTS 71
C. COLOR AND FILL 73

APPENDIX A CHART AND SKETCH USER'S MANUAL 75

APPENDIX B DOCUMENTATION OF CSC ROUTINES 123

APPENDIX C HARDWARE AND SOFTWARE REQUIREMENTS 137

LIST OF REFERENCES 143

INITIAL DISTRIBUTION LIST 145

I. BACKGROUND

Chart and Sketch was initially written by CDR Gary Porter in 1979 using the GL2 graphics programming language, designed for use on the Tektronix 4014-1 Graphics Terminal.

The goal of this thesis was to develop an interactive graphics program which incorporated the basic capabilities of the aforementioned program as well as the enhanced graphics capabilities available through the DI-3000 software and the RAMTEK Graphic Display System, including color selection, zoom and panning capabilities, fill and text options. The program is resident on the VAX 11-780 in the Wargaming Analysis and Research Laboratory (WARLAB) at the Naval Postgraduate School, Monterey, CA (NPS). It executes under the VMS operating system. A detailed User's Manual has been written and is attached as Appendix A to this thesis. The user's manual is also available on line in the WARLAB.

Chapter II presents a detailed description of the programming concepts used in the construction of Chart and Sketch. Algorithms and narrative are used to explain the design of Chart and Sketch and relate its design to the programming concepts.

Chapter III examines the basic graphics capabilities used by Chart and Sketch, their implementation and their integration with the principles of design. Alternative implementations are also examined.

Chapter IV discusses the possible applications of Chart and Sketch in the Command and Control environment, as a reporting and briefing tool, and as a general graphics tool. Chapter V addresses the hardware, software and physical prerequisites to successfully adapting Chart and Sketch to various computer and graphics systems.

II. PROGRAM DESCRIPTION

The purpose of Chart and Sketch is to allow a novice graphics user to effectively utilize the graphics capabilites of the DI-3000 software and the RAMTEK RM-9460 Graphic Display System without any knowlege of FORTRAN or DI-3000 programming. At this time, the user is encouraged to turn to the User's Guide (Appendix A) to obtain a detailed operational understanding of the capabilities and limitations of Chart and Sketch.

Chart and Sketch is a highly interactive, color, computer graphics program. It is a Fortran 77 implementation of the Precision Visual DI-3000 graphics software, the CSC (Conflict Simulation Center, Lawrence Livermoore, CA) enhancements to the DI-3000 software and the RAMTEK RM-9460 Series Graphic Display System hardware.

DI-3000 is an integrated system of device independent, graphics software tools that may be called as subroutines in a FORTRAN program. These subroutines are device-independent. The device driver of the DI-3000 interprets these device-independent graphics commands and issues RM-9460, device-dependent graphics commands allowing the implementation program to interact with the RM-9460 Graphic Display System [Ref. 1]. The CSC implementation of

the DI-3000 inludes numerous, additional graphics routines which enhance the DI-3000 graphics capabilities [Ref. 2]. These routines are documented in appendix B to this thesis.

The RAMTEK RM-9460 is an intelligent graphics display system used in conjuction with the DI-3000 software to produce graphics products. The system consists of a graphics processor with refresh memory and one or more raster color monitors. It is capable of both multiview and interactive operation. [Ref. 3]

Chart and Sketch is a menu driven, highly interactive program. Each of the menus is presented in a logical, hierarchial process which makes the creation of graphics segments as simple and straightforward as possible. A graphics segment is a collection of graphics commands. A segment may be thought of as image created by a collection of graphics commands is retained in memory by the DI-3000. The display on the RAMTEK RM-9460 graphics screen is made up of one or more of these segments.

Chart and Sketch creates graphics segments for the user automatically. These segments can be moved, deleted, copied, or magnified by the user without any knowledge of the process involved. The results of any graphics session, (the image on the graphics monitor) may be saved as a file in any VMS directory and can be retrieved (redisplayed on the graphics monitor) at a later date using Chart and Sketch.

A. PROGRAM DESIGN CONCEPTS

This section describes a number of the more important graphics design concepts which were employed in the creation of Chart and Sketch. The principles of design were drawn from numerous sources, but special credit is afforded Foley and Dam [Ref. 4].

1. Simplicity

The first principle considered in the design of Chart and Sketch was that of simplicity from the view of the user. A major goal of Chart and Sketch was to allow the most novice user to make full use of the capabilites of color graphics with a bare minimum of training. Since a program is usually easier to use if menus are provided [Ref. 5], Chart and Sketch was designed around menu selection.

To the extent possible, all user interaction occurs by menu selection at the graphics screen through use of a graphics tablet (as a locator), and a cross-hair cursor (as a pick device). The exceptions require input from an alphanumeric terminal keyboard. Keyboard input is usually required for character strings although it is sometimes employed to reduce alternation between the graphics screen and the terminal keyboard.

2. Minimization of Memorization

The minimization of memorization was another principle of design. Learning to use any new system or program requires some degree of memorization. The less

there is to memorize, the more quickly the system can be learned. Chart and Sketch limits the memorization required to learn the program by employing a generous number of user prompts, consistency, and menu selection.

User prompts are provided both on the graphics screen and on the alphanumeric terminal before each user input. Where input is required at the alphanumeric terminal keyboard, prompts appear on both the graphics screen and alphanumeric terminal screen. Many of the prompts provide more detailed instructions when the user inputs a "?".

3. Consistency

Consistency was also a prime consideration in the design of Chart and Sketch. The physical location of prompts and menus on the graphics screen remains consistant throughout the program. The way in which menu options are selected and implemented is also consistent. In a few instances it was deemed desirable to provide additional means of implementation; however, in such instances, the additional implementation is purely optional.

4. Feedback

The fourth principle of design was feedback, both syntactic and semantic. Syntactic feedback occurs when a unit (word) of the input language is accepted by the system. Its purpose is to inform the user that the input has been accepted. Chart and Sketch provides syntactic feedback on both the graphics and alphanumeric terminal screens. When a

menu selection is accepted by the program either the menu from which the selection was made disappears and/or a new prompt is displayed. The user receives immediate feedback. Syntactic feedback is in the form of another query, prompt, or a request to STANDBY.

Semantic feedback informs the user that the requested action has been compeleted. In Chart and Sketch, semantic feedback usually consists of a modified display on the graphics screen which shows the results of the requested action. When this is not the case, (eg: when working at the alphanumeric terminal), semantic feedback is provided through messages on the alphanumeric terminal screen. In some cases, when a command cannot be executed quickly, the user will receive semantic feedback in both forms, ie: a message explaining what the program is doing as it is processing the command, and a graphic display when the command is completed. (This occurs, for example, when the user creates a section file.)

5. Error Accommodation

The fifth principle of design involved the accommodation of human errors. It is common to make mistakes while working with any program, regardless of the user's level of expertise. A user friendly program should provide an easy way to recover from inadvertant mistakes without imposing harsh consequences on the user. Chart and Sketch provides two useful ways to recover from mistakes.

It is not an uncommon mistake to accidentally delete or erase important information. Since Chart and Sketch provides the user the capability to delete all graphics on the graphics screen (either by selecting CLEAR or QUIT from the Main menu), it also provides a safeguard. When either CLEAR or QUIT is selected from the Main menu, a second menu appears with the options "yes" and "no", and the user is required to verify the menu selection before it is implemented by the program. If "no" is selcted the user is returned to his previous place in the program and no graphics are lost.

Another error accommodation is provided in a routine called DELETE. Delete allows the user to selectively delete images on the graphics screen. If a mistake is made in creating, recalling, or otherwise adding a graphic image to the screen, the image can be deleted without effecting any of the other images on the screen (ie: the user can return to where he was before he made the mistake).

6. Response Time

The sixth, and final design principle, involved control of response time. Miller [Ref. 6] has proposed that there exists a hierarchy of required response times that is related to the psychological feeling of "closure". He postulates that a larger task has a greater tolerable response time, while a smaller task has a smaller acceptable response time. Foley [Ref. 4] points out that a system

pays two costs for response times that are perceived by the user as delays: wasted user time and loss of the user's train of thought. Chart and Sketch was designed to minimize response time by delaying certain time consuming graphics processes and by using the alphanumeric terminal (vice the graphics screen) to display time consuming feedback which might be useful in some applications, but is not essential to any.

The graphics operation which requires the greatest amount of time is that of erasing and redrawing the entire graphics screen. This is necessary in some instances to completely erase deleted graphic images. In order to minimize the number of times the screen has to be erased and the graphics redrawn, those routines which utilize the deletion of graphic images (DELETE, MOVE, and COPY), are presented to the user as batch operations. That is, the user may delete, move, or copy multiple images before the screen is erased and the graphics redrawn to "clean up" the picture. The user is, however, provided the ability to force a new screen and redrawing of the graphics at any time during the batch process.

The following section will examine the the basic graphics capabilities of Chart and Sketch.

B. PROGRAM ALGORITHM

This section presents the basic algorithm used in the design of Chart and Sketch. The algorithm is written using a high level of abstraction and subroutines in order to present the reader with a simple and yet complete overview of the program design. The algorithmic language and format are from Graham [Ref. 7].

1. Main Program Algorithm

```
ALGORITHM MAIN
{The Main program processes a menu selection
and calls other procedures.   It initializes
and ends the program.}

    CALL subroutine  Initz
    {Initz initializes program variables, prompts,
    and menus.}

    CALL subroutine Menu
    {Menu creates the prompts and menus.}

    REPEAT
        post the Main menu

        get user selection from Main menu

        {Main menu options include: chart, sketch,
        retrieve, file, clear, view, move, and
        quit.}

        IF menu selection = chart THEN
            CALL subroutine wdata

        ELSE IF menu selection = sketch THEN
            CALL subroutine Draw

        ELSE IF menu selection = retrieve THEN
            get file name and type
            IF filetype = data THEN
                CALL subroutine Retrieve_data
            ELSE IF filetype = section THEN
                CALL subroutine Retrieve_sec
            END IF

        ELSE IF menu selection = file THEN
```

```
            get file name and type
            IF filetype = data THEN
                CALL subroutine File_data
            ELSE IF filetype = section THEN
                CALL subroutine File_sec
            END IF

        ELSE IF menu selection = clear THEN
            verify menu selection
            delete all graphics
            reintitialize graphics variables

        ELSE IF menu selection = view THEN
            redraw graphics to fill the entire screen
            redraw to normal size at user request

        ELSE IF menu selection = move THEN
            get the segment number
            get the new location
            erase the old segment
            redraw segment in the new location

        ELSE IF menu selection = zoom THEN
            magnify the graphics by a power of 2
            allow user to pan using locator device
            return to normal view at user request

        ELSE IF menu selection = quit THEN
            verify the selection
            quit <-- true

        END IF
    UNTIL quit = true

END MAIN
```

The Main program of Chart and Sketch is designed as a loop which uses menu selections to invoke program functions. In most instances, further user input through sub menu selection is required. The Main program calls subroutines to invoke many of the menu options. The segmentation of the program into subroutines has two purposes: to facilitate program design and to provide for recovery from errors.

The DI-3000 allows recovery from "fatal" errors which occur in a subroutine by returning to the Main program at the point following the call to the subroutine. By using subroutines to invoke complex and inherantly error-prone program functions, recovery from "fatal" errors is facilitated.

The following flowchart diagrams the hierarchy of menus as presented by Chart and Sketch.

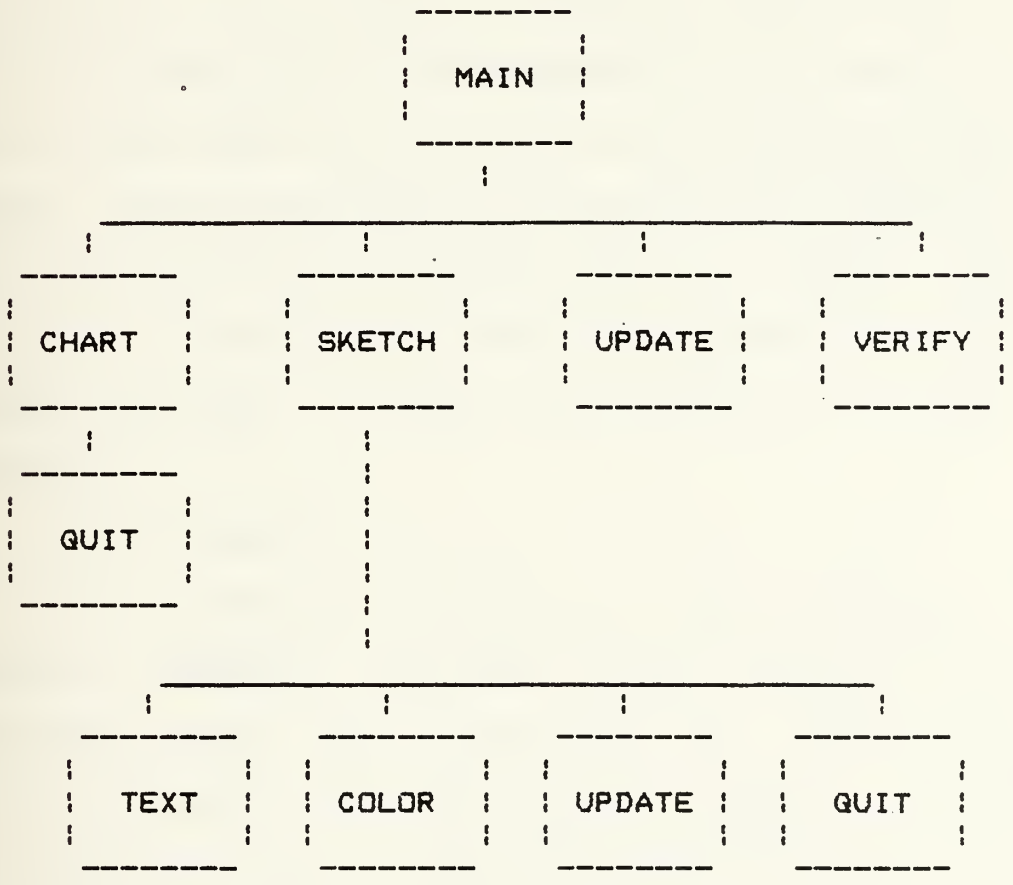

Figure 1. Menu Hierarchy

The options which the user may select from the main menu include the following.

a. Chart

If CHART is selected, the user is presented another menu from which to select a more specific action. To implement this option the main program calls the subroutine Wdata. Chart and Sketch was designed to provide for easy modification in the event that an external database for drawing charts becomes available. All the procedures involved in the creation or reading of charts are invoked from the Chart Menu. The program functions that are unique to the Chart Menu are provided by the subroutine Wdata. No other program functions or subroutines are dependent on the functions provided by the Chart Menu or Wdata subroutine. Only the Chart Menu and Wdata subroutines would have to be modified to add a routine to access an external chart database.

b. Sketch

Sketch is selected to draw, delete, or copy graphics segments, including circles, diamonds, rectangles, polygons, lines, arcs, sectors and text. Sketch allows the user to select the color and fill attibute. If SKETCH is selected, the user is presented another menu from which to select a more specific action. To implement the Sketch option, the main program calls the subroutine Draw which calls Sketch, Filclr, Text, Dup, Delete, and Polygon.

c. File

Is used to save a currently displayed graphics product by encoding the graphics data and writing it to a file in the user's directory. This option calls either the subroutine File_data or File_sec.

d. Retrieve

Retrieve is used to redisplay a previously created graphics product from a file in the user's directory. This option calls either the subroutine Retrieve_data or Retrieve_sec.

e. Clear

Clear is selected to delete all previous work done during the session which has not been saved to a file. This option is implemented by the main program without a call to any subroutines.

f. View

View redraws the currently displayed graphics product to fill the entire monitor screen. If VIEW is selected, the action is executed by the main program.

g. Move

Move changes the location of a displayed segment on the screen. The main menu will disappear and a prompt for appropriate user action will appear. The main program also implements this option with calling any subroutines.

h. Zoom

Zoom magnifies the currently displayed graphics by a factor of 2 and allows the user to interactively change the reference center (pan). No subroutines are called to implement this feature.

i. Quit

Select QUIT to exit the program. The program will end after the user has verified his intention to exit the program.

2. Subroutine Algorithms

The Main program of Chart and Sketch is heavily dependent upon subroutines to implement most of the graphics options presented the user. The subroutines were carefully designed to provide specific functions. In several instances, the functions provided are used by other subroutines as well as the Main program. In this section the algorithms used in the design of the subroutines are presented and discussed.

a. Subroutine Initz

```
{Initz  initializes the viewing and port parameters,
assigns logical device, prompt, and menu names, and
initializes other program variables.}

    get the number of monitor pairs to use
    get the names of the monitor pairs
    initialize the selected monitors
    define the drawing area viewport parameters
    define the menu viewport parameters
    define the drawing area window parameters
    define the menu window parameters
    define background, menu, and prompt colors
    assign integer names to the menus
```

```
    assign integer names to the prompts
    define DI-3000 graphics default values

END INITZ
```

Subroutine Initz is called by the main program. It is used to initialize various aspects of Chart and Sketch including, the user selected monitors, viewport and window parameters, menu and prompt names and DI-3000 default values.

Initz prompts the user to enter an integer number representing the graphics monitors to be initialized and used by Chart and Sketch. Initz then uses DI-3000 routines to initialize the selected monitors.

The viewport parameters identify a virtual rectangular area of the graphics screen. The window parameters define a world rectangle which is then mapped , along with its graphics contents, onto an associated viewport on the screen. By defining multiple viewports and windows the implementation program can confine the display to selected portion of the graphics screen.

The contents of a segment (a collection of graphics commands) are assigned integer names by Initz (menu and prompt names). These integer names allow the application program to modify global attributes of the menus and prompts, such as visibility and detectability (discussed later in this thesis).

Consistency is achieved by assigning the same viewport parameters to all menu segments, thus confining the presentation of all program menus to the same physical location on the graphics screen. In like manner, the program prompts and drawing area are assigned physical locations on the screen.

The values of the viewports, windows, menus and prompts are all parameterized and placed in common to provide for easy program modification. The Menu subroutine uses the viewport, window, menu and prompt parameters in the creation of the program menus and prompts.

b. Subroutine Menu

```
{Menu creates the menus and prompts used
by the program.}

    set the viewport and window for the menus
    create the menus as invisible segments
    set the viewport and window for the prompts
    create the prompts as invisible segments

END MENU
```

The main program calls the Menu subroutine to create the program menus and prompts. The menus and prompts are created invisibly, within the viewports and windows assigned by the subroutine Initz. The menus and prompts are made visible (displayed on the graphics screen) by the program.

c. Subroutine Draw

```
{Draw presents the Sketch Menu, processes the
user selection from the menu, prompts the user
for required input, and calls subroutine sketch
to create the graphics segment defined by the
user's inputs.}

   set the viewport and window for the drawing area

   REPEAT

      post the Sketch menu
      {options include: rectangle, diamond, circle,
      polygon, line, arc/sec, text, color, fill, delete,
      copy, and quit.}

      get user selection from the Sketch menu

      IF menu selection = rectangle THEN
          get the coordinates of two diagonal points
           on the rectangle
          CALL subroutine Sketch to create the rectangle

      ELSE IF menu selection = diamond THEN
          get the coordinates of the center and
           radius of the diamond
          CALL subroutine Sketch to create the diamond

      ELSE IF menu selection = circle THEN
          get the coordinates of the center and
           radius of the circle
          CALL subroutine Sketch to create the circle

      ELSE IF menu selection = polygon THEN
          CALL subroutine polygon to get the coordinates
           of up to 100 points that define the polygon
          CALL subroutine Sketch to create the polygon

      ELSE IF menu selection = line THEN
          CALL subroutine line to get the coordinates
           of any number of points that define a polyline
          CALL subroutine Sketch to create the polyline

      ELSE IF menu selection = arc/sec THEN
          get the coordinates of the center of the
           subtended circle
          get the coordinates of a line which defines
           the starting angle of an arc
          get the coordinates of a line which defines
           the ending angle of an arc
          get the coordinates of a point which defines
```

```
            the radius of the subtended circle
            IF the current fill attribute = filled THEN
                CALL subroutine Sketch to create an arc
            ELSE
                CALL subroutine Sketch to create a sector
            END IF


        ELSE IF menu selection = text THEN
            get the coordinates of the leftmost character
            CALL subroutine Text to get character font/size
            get the text string
            CALL subroutine Sketch to create the text string

        ELSE IF menu selection = color THEN
            CALL subroutine Filclr to get the new color
            current color <-- new color
            post the Sketch menu in the current color

        ELSE IF menu selection = fill THEN
            IF the current fill attribute = hollow THEN
                the new fill attribute <-- solid
            ELSE
                the new fill attribute <-- hollow
            END IF
            post the Sketch menu using the new fill attribute

        ELSE IF menu selection = delete THEN
            CALL subroutine delete to delete segments

        ELSE IF menu selection = copy THEN
            post the copy menu
            {options include: quit and update}
            get the number of the segment to be copied
            get coordinates of the new location
            CALL subroutine Dup to make copy at new location
            make new segment visible

        ELSE IF menu selection = quit THEN
            quit <-- true

        END IF

    UNTIL quit = true

END DRAW
```

The Draw subroutine presents the prompts and menus necessary for the interactive selection and definition of graphic segments. The Draw subroutine presents the user with the Sketch menu options, processes the selection, and then prompts the user for the input required to create the selected graphics segment. Selections from the Sketch menu identify the primitive (rectangle, diamond, circle, polygon, line, arc/sec, text), color (red, green, yellow, blue, magneta, cyan, white, none), and interior style (filled or hollow) attributes of the graphics objects in the segment to be created. It also allows previously created segments to be copied or deleted. Options from the Sketch menu include:

RECTANGLE, to create a horizontally or vertically oriented rectangle. This option allows the user to draw a rectangle of any height and width using the current color and fill characterstics.

DIAMOND, to create a diamond with points oriented at the cardinal headings. This option allows the user to draw a diamond with any center and any size using the current color and fill characterstics.

CIRCLE, to draw a circle. This option allows the user to draw a circle with any center and radius using the current color and fill characterstics.

POLYGONS, to draw one or more polygons of up to one hundred points each. This option allows the user to draw multiple polygons, using the current color and fill

characterstics. To implement this option, Draw calls the Polygon subroutine.

LINES, to draw one or more lines. The user may draw multiple polylines in the current color. To implement this option, Draw calls the Polyline subroutine.

ARC/SEC, to draw an arc or a sector of any size using the current color characteristic. If the current fill characteristic is "filled" a filled sector will be created, otherwise an arc will be drawn.

TEXT, to allow the user to place a text string on the graphics screen in any location using the current color characteristic. Draw calls the Text subroutine to determine the style and size of font to be used.

COLORS, to change the current color characteristic. Draw calls the Filclr subroutine to allow the user to select the color characteristic. Once a segment has been drawn its color cannot be changed. The color attribute must be selected before the segment is created.

FILL, to change the current fill characteristic. The FILL option on the SKETCH menu controls the fill attribute used in drawing graphics segments. Selecting the FILL option toggles the fill attribute between HOLLOW and FILLED.

DELETE, to to selectivly delete (erase) any graphics segment drawn using SKETCH. Draw calls the Delete subroutine to implement this option.

COPY, to make copies of graphics segments which have already been drawn. A copy of the selected segment is drawn at another location using the currently selected color attribute.

QUIT, to return to the main program menu. No data is lost in returning to the MAIN menu. If QUIT is accidently selected from the SKETCH menu, the user can return to exactly the same place in the program by reselecting SKETCH from the MAIN menu.

The Draw subroutine calls for and stores the points which define the selected segment. It also provides for the changing of the current fill characteristic. If polygon or line is selected, a subroutine is used to call for and store the defining points (subroutines Polygon or Line repectively). Draw calls the subroutines Filclr and Text to define the current drawing color and text font. Subroutine Delete is called to delete segments and subroutine Dup is called to duplicate segments.

After the color, fill, and defining points are stored, Draw calls the Sketch subroutine to create the segment. Sketch is not incorporated in Draw because the functions provided by Sketch are also required by the subroutines Retrieve_data and Wdata.

d. Subroutine Filclr

```
{Filclr lets the user select a color to sketch in.}

    post the color menu
    get the user selection

END FILCLR
```

Draw calls the Filclr subroutine to provide the user with the option of changing the color in which subsequent segments are drawn. It presents the color menu, which displays all eight available colors, accepts the user's menu selection, and returns the value of that selection to the Draw subroutine.

e. SUBROUTINE TEXT

```
{Text lets the user selet a font and size for text.}

    post the Text menu
    get the user's selection
    assign a font and size value to the selection

END TEXT
```

Draw calls the Text subroutine to allow the user to select a size and font (style) for writing text on the graphics screen. Text presents the Text menu, displaying various sizes and fonts, accepts the user's menu selection, and returns a font value to the Draw subroutine.

f. Subroutine Sketch

```
{Sketch numbers, records, and creates graphics
segments.}

    assign the new segment a number
```

```
    IF only 5 segments are left to be assigned THEN
        warn the user
    ELSE IF there are no segments left THEN
        inform the user that no segments are left
        RETURN to the calling routine
    END IF

    store the segment attributes for later recall
    create the segment using DI-3000 routines
    make the segment detectable
    make the segment visible

END SKETCH
```

The Sketch subroutine provides the basic interface between the implementation program and the DI-3000 graphics routines. The design principle, Minimization of Memorization, finds fulfillment in the Sketch subroutine. The user does not need to remember any DI-3000 graphics commands or parameter list formats to create a graphics segment.

Prior to creating the segment, Sketch assigns a number to be used in identifing the segment. This number is then used by the Delete and Dup subroutines and the main program to delete, copy or move the segment.

Chart and Sketch provides the user with the facility to display up to 1000 segments simultaneously. If there are five or less segments left, Sketch warns the user with a message and a bell tone at the alphanumeric terminal, and then creates the segment. If there are no segments left, Sketch warns the user with a message and bell at the alphnumeric terminal, but does NOT create the segment.

The Sketch menu selection, which defines the type of graphics segment, and the parameters required to create the segment, are passed to the Sketch subroutine by the calling routine. Sketch formats the parameters and calls the DI-3000 routines required to create the segment. Sketch also makes the segment detectable and visible, and stores the segment's parameters for later recall by the subroutines File_data, Dup, and Move. The user does not need to remember the segments's parameters to duplicate or move it. Chart and Sketch "remembers" for the user.

g. SUBROUTINE CLEAR

```
{Clear deletes all graphics segments and
reinitializes graphics variables.}

    IF no segments exist THEN
       RETURN to the Main program

    ELSE
       purge all user created segments from memory
       all segment visibility variables <-- false
       segment numbers <-- initial values

    END IF
END CLEAR
```

The Clear subroutine, called by the main program, deletes (purges) all segments created by the user. It provides the facility to effectively "restart" the program. Its intended use is to start another graphics project after the first has been completed and stored in a file, or to restart an unsuccessful graphics session (a global error accommodation).

h. Subroutine File_sec

```
{File_sec encodes the contents of the graphics
memory planes (pixel data) and writes it to a
section file.}

    ensure that the filetype is .SEC
    rewrite the graphics to fill the entire screen
    get the window size using KSBYTE24
    create the section file using KSCRET
    encode/write the bit planes using JESCAPE 9425
    copy the section file to the user's disk
    delete the section file from virtual memory
    deassign the section
    close the file
    rewrite graphics to their original size

END FILE_SEC
```

i. Subroutine File_data

```
{File_data writes the parameters of each visible
segment (stored by subroutine Sketch)  to a data
file in the user's directory.}

    open the file

    DO FOR segment = first segment TO last segment
       IF the segment is visible THEN
           write it's attributes to the data file
       END IF
    END DO

    close the file

END FILE_DATA
```

The subroutines File_sec and File_data, called by the main program, encode the results of a graphics session (the picture on the graphics screen) and store it in a file. The first method, used by File_sec, encodes the memory planes (pixel data) used to project the image onto the graphics screen using a CSC routine. Since it creates a

pixel image (one byte per pixel), it uses large amounts of storage, but was deemed necessary because it provides interprogram flexibility. A file created by File_sec can be read and redisplayed by any program having access to the CSC enhanced DI-3000 routines.

The second method, File_data, encodes the segment parameters passed to the subroutine Sketch (which were used to create the segments). The advantages and disadvantages of File_sec verses File_data will be discussed at length later in this thesis.

The KSBYTE24, KSECRET, and JESCAPE 9425 routines are all documented in appendix B to this thesis. KSBYTE24 and KSECRET are routines added by CSC. The JESCAPE 9425 is an escape code which directly controls the pixel data transfer between the RM-9460 memory planes and the host VAX.

j. Subroutine Retrieve_sec

```
{Retrieve_sec reads a section file from the user's
directory and writes it on the graphics screen.}

    REPEAT
        get the name of the file
        ensure that the filetype is .SEC
        ensure that the file exists
        open the section using KSOPEN
        set the viewport and window to use the entire screen
        read the file using JESCAP 9424
        deassign the section
        close the file
        see if the user wants to read another section file
    UNTIL the user doesn't want to read another section file

END RETRIEVE_SEC
```

k. Subroutine Retrieve_data

```
{Retrieve_data reads a user data file and recreates
graphics by calling the subroutine Sketch.}

    ask the user if he wants to use the full viewing space

    IF the user doesn't select full viewing space THEN
        get the coordinates of the viewspace to be used

        REPEAT
            read a graphics command from the data file
            scale the data coordinates to new viewspace
            CALL subroutine Sketch to create the segment
        UNTIL the end of the file is reached

     ELSE IF the user selects full viewing space THEN

        REPEAT
            read a graphics command from the data file
            CALL subroutine Sketch to create the segment
        UNTIL the end of the file is reached

      END IF

END RETRIEVE_DATA
```

The main program calls the Retrieve_sec and Retrieve_data subroutines to provide the facility for reading and redisplaying the files created by File_sec and File_data. The JESCAPE 9424 routine is documented in Appendix B to this thesis.

Retrieve_data and Retrieve_sec each provide a unique facility. Retrieve_data provides the user the ability to redisplay a file in a reduced size, anywhere in the drawing area. To do this, the user is prompted to enter two points defining a rectangle.

The vertical aspect of the rectangle is adjusted to prevent any distortion of the original display. This adjusted rectangle is then used as the viewport in redisplaying the selected file.

Retrieve_sec allows the user to redisplay multiple files without reselecting the retrieve option each time from the main menu. This is implemented by introducing a loop in the Retrieve_sec algorithm.

1. Procedure Points

```
{Points accepts user input (coordinate values) from the
locator, then calls the Subroutine Sketch to create the
graphics segment defined by the points.}

    present the user with instructions
    post the quit menu
    REPEAT
        ask for a locator input
        IF the locator input = quit THEN
           IF a graphics segment has been defined THEN
               CALL subroutine Sketch to create the segment
           END IF
           quit <-- true
        ELSE IF the locator input not = quit THEN
           accept the locator input as coordinate values
           IF the button depress coordinates =
                     the button release corrdinates THEN
              add the coordinates to the current segment
           ELSE
              end the current segment input
              CALL subroutine Sketch to create the segment
              initialize a new segment
           END IF
        END IF
    UNTIL quit = true
END POINTS
```

The POINTS algorithm reflects the design of several subroutines, including: Line, Polygon, and Wdata. (Line and Polygon are called by the Draw subroutine, while

Wdata is called by the main program.) The basic design of each subroutine was similar, but the user prompts and menus were different. The routine accepts multiple locator input coordinates.

Each pair of coordinates, representing a point on the graphics screen, is tested to see if it represents a menu selection; if it does, the menu function is invoked. If it does not represent a menu selection it is accepted as the point coordinates of a polyline or polygon.

This test could have been eliminated by requiring the user to input menu selections at the VT100/102 alphanumeric terminal, but simplicity (from the user's point of view) is served when all input is confined to one device.

Furthermore, the coordinates returned to the program during pick button depression and pick button release are compared. If they are equal, the point is accepted as an additional point of the current polyline or polygon. If the coordinates are different, the point is accepted as the first point of a new polyline or polygon.

This provides the user the ability to draw multiple polylines or polygons without reselecting the option from the Sketch menu. This procedure does NOT conflict with the design principle of consistency. Although it provides the additional capability of drawing multiple segments with just one menu selection, it does NOT prevent the user from using the usual method.

More specific information concerning the design of Chart and Sketch is available by reference to the source program. The source program for Chart and Sketch is available in the Wargaming Analysis and Research Laboratory (WARLAB) at the Naval Postgraduate School, Monterey, CA.

III. PROGRAM IMPLEMENTATION

This section examines the the basic graphics capabilities used by Chart and Sketch, their implementation and their integration with the aforementioned principles of design, ie: simplicity, minimization of memorization, consistency, feedback, error accommodation, and control of response time. Both the method of implementation used and alternate possibilites will be examined.

A. USING MENUS TO INVOKE FUNCTIONS

Chart and Sketch is designed as a menu driven, highly interactive program designed to provide simplicity of execution and immediate user feedback. In order to avoid dividing the user's attention between the alphanumeric terminal and screen and the graphics screen and locator, Chart and Sketch is written to maximize the amount of user interaction at the graphics screen and to minimize interaction at the alphanumeric terminal. This is a design decision which takes into account both the advantages and disadvantages of presenting menus and prompts on the graphics screen.

The primary disadvantage is that the menus and prompts use up valuable screen area that might otherwise be used for graphics display. However, this diadvantage is outweighed by the very tangible benefits realized when the user does not have to divide his attention between the graphics monitor and the alphanumeric terminal. Both simplicity and consistency are served by conducting as much of the interaction as possible at one physical location.

The menus are always presented on the right side of the graphics screen within a vertically oriented rectangle. The options are displayed as text strings or graphic symbols inside the rectangle.

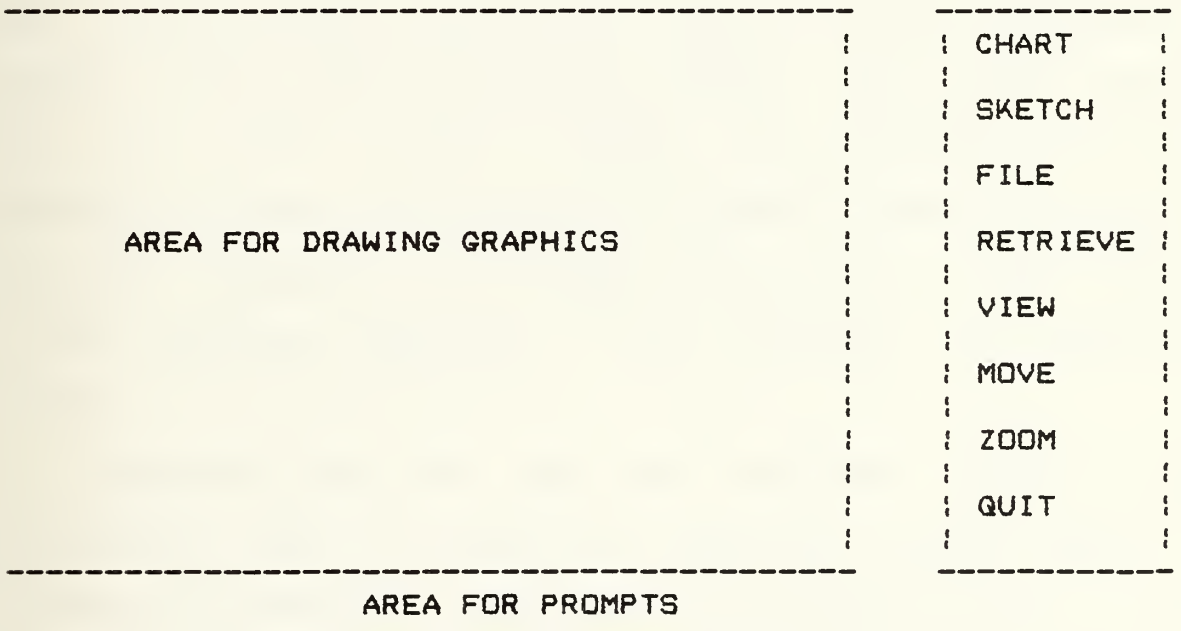

Figure 2. Main Menu

Each menu is defined as a separate graphic segment (image) by the DI-3000 JROPEN and JCLOSE routines. Within the segment, each text string or symbol is defined by the

DI-3000 JPKID routine. JROPEN assigns a unique name (in the form of an integer) to each segment when it is created. JPKID assigns an integer value to each text string or symbol that appears in the menu. A third DI-3000 routine, JPICK, is used in conjuction with the locator to return this integer value (of a selected menu option) to the implementation program.

The tablet locator is a common graphics input device used to specify screen coordinates. When the locator is moved over the surface of the tablet, its position on the tablet is echoed on the graphics screen as a cursor. The cursor provides immediate, semeantic feedback. In moving the locator across the graphics tablet, the user requests a change in the cursor coordinate location. The cursor echo on the graphics screen reflects the completion of the requested action. When a button on the locator is depressed (and or released), the screen coordinates of the current cursor location are sent to the computer or interface device.

Selection from the menu was designed around the interactive use of locator/pick devices. A menu option is selected by moving the pick across the graphics tablet (the locator) until the screen cursor is positioned over the center of the desired option and then pressing a button on the pick device. The locator then returns coordinates of the cursor to the application program.

Function invocation from the menu is achieved by correlating the coordinates returned by the locator with the coordinate positions associated with the menu options. If the cursor coordinates are within a given radius of a menu option, the DI-3000 will return the integer value assigned that option (using JPKID) to the application program for processing. This integer value can then be used by the application program to invoke an associated function.

The following two examples from Chart and Sketch illustrate the creation of a menu segment (using JROPEN, JCLOSE and JPKID), and a program call to return the JPKID value of the selected menu option. Error accommodation is provided in the menu selection procedure. An invalid JPKID value has no effect. (See Ref. 1 for a detailed description of the DI-3000 routines.)

CREATING A MENU

```
C --- MAKE MENU 1, THE MAIN MENU
      CALL JROPEN (MENUS(1))       !BEGIN MENU
        CALL JPINTR(1)             !define fill attribute
        CALL JCOLOR (MCOLOR)       !define color
        CALL JJUST (1,2)           !left justify options

C --- WRITE MENU 1
        CALL JSIZE (2.5,2.5)       !define character size
        CALL JPKID (1)             !SET PICK VALUE= 1
        CALL JMOVE (-8., -5.)      !start text here
        CALL J1TEXT (5,5HCHART)    !OPTION 1
        CALL JPKID (2)             !SET PICK VALUE= 2
        CALL JMOVE (-8.,-11.)
        CALL J1TEXT (6,6HSKETCH)   !OPTION 2
        CALL JPKID (3)             !SET PICK VALUE= 3
        CALL JMOVE (-8.,-17.)
        CALL JSIZE (2.,2.5)
        CALL J1TEXT (8,8HRETRIEVE) !OPTION 3
        CALL JPKID (4)               !SET PICK VALUE= 4
        CALL JMOVE (-8.,-23.)
```

```
      CALL JSIZE (2.5,2.5)
      CALL J1TEXT (4,4HFILE)     !OPTION 4
      CALL JPKID (5)             !SET PICK VALUE= 5
      CALL JMOVE (-8.,-29.)
      CALL J1TEXT (5,5HCLEAR)    !OPTION 5
      CALL JPKID (6)             !SET PICK VALUE= 6
      CALL JMOVE (-8.,-35.)
      CALL J1TEXT (4,4HVIEW)     !OPTION 6
      CALL JPKID (7)             !SET PICK VALUE= 7
      CALL JMOVE (-8.,-41.)
      CALL J1TEXT (4,4HMOVE)     !OPTION 7
      CALL JPKID (8)             !SET PICK VALUE= 8
      CALL JMOVE (-8.,-47.)
      CALL J1TEXT (4,4HZOOM)     !OPTION 8
      CALL JPKID (9)             !SET PICK VALUE= 9
      CALL JCOLOR (6)
      CALL JMOVE (-8.,-54.)
      CALL J1TEXT (4,4HQUIT)     !OPTION 9

  CALL JRCLOS                    !END OF MENU
```

MENU SELECTION

```
C --- PROMPT THE USER TO PICK FROM THE MENU

C --- READ THE USER'S SELECTION (SELECTION = IPKID)
10    CONTINUE
      CALL JPICK(1,1,1,IBUT,ISEG,IPKID)
      IF (IPKID .LT. 1 ) GOTO 10      !error accommodation

C --- PROCESS SELECTION
      GOTO (11, 12, 13, 14, 15, 16, 17, 18, 90), IPKID
```

The first five lines of the above algorithm define the fill, color and justification attributes to be used in writing the menu options. The remaining lines define the size of the text to be used, the value to be returned if the following option is selected, and the coordinate location where the menu option is written. The size of the text is changed so all text strings fit within the menu.

The routine JPICK assigns an integer value to the variable IPKID corresponding to the option selected by the user. The GOTO statement uses this value of IPKID to invoke the desired graphics function. The application program provides syntactic feedback by presenting the user with a new menu or prompt.

B. MAKING SEGMENTS VISIBLE AND INVISIBLE

As previously defined, a segment is a collection of graphics commands which results in a single graphics image. DI-3000 allows the implementation program to control the visibility of a segment as "visibile" or "invisible". Making segments visible is called "posting" , while making them invisible is called "unposting". Chart and Sketch implements this capability in several contexts. The first to be examined will be its use in presenting menus and prompts.

All Chart and Sketch menus and prompts are initially created invisibly. Although they exist in memory from the time they are created, they are not posted (made visible) until the application program changes their visibility attribute to "visible". When the menu or prompt is no longer appropriate, it can be unposted (deleted) from the graphics screen by again changing its visibility, this time to "invisible". Posting provides a very convenient way to display or erase program menus and prompts.

Chart and Sketch uses the DI-3000 routine JVISBL to control the visibility of menu and prompt segments.

POSTING MENUS AND PROMPTS

```
75    CONTINUE
         CALL JVISBL (MENUS (1), 0) !make menu 1 invisible
         CALL JVISBL (PROMPT(1), 0) !make prompt 1 invisible
         CALL JVISBL (MENUS (9), 1) !make menu 9 visible
         CALL JVISBL (PROMPT(14),1) !make prompt 14 visible
```

Chart and Sketch allows the user to delete and move graphics segments (a form of error accommodation). Since moving a segment actually consists of deleting the segment and recreating it in a new location, both deleting and moving segments requires the ability to control the segment's visibility. Chart and Sketch deletes a segment by setting its visibility attribute to "invisible".

The DI-3000 routine JPICK can be used to return the integer name of any visible segment on the graphics screen in the same way it returns the PICK value of a menu option. By using JPICK to identify a visible segment, an important capability is realized: the user can interactively identify any visible segment. Minimization of memorization is achieved as the user does not have to remember any segment's integer name.

In the following example, the user is prompted to use the locator to select a segment (to be deleted) or select an option from the menu. (The options are: quit and update). The user positions the cursor over a segment or a menu

option and depresses a button on the locator. JPICK is called to return either a segment value or an option value. If a menu value is returned, the option represented by that value is executed, (either the screen is updated or the user is returned to the Sketch menu). If a segment value is returned, the segment represented by that value is deleted from the screen by changing the segment's visibilty to "invisible" (discussed later in this thesis).

DELETING SEGMENTS

```
C --- MAKE THE DELETE MENU VISIBLE
      CALL JVISBL (MENUS(9), 1)

10    CONTINUE
C --- PROMPT THE USER TO START DELETING
      CALL JVISBL(PROMPT(4),1)  !delete prompt

15    CONTINUE
         CALL JPICK (1, 1, 1, IBUT, ISEG, IPKID)

                    !IPKID is the menu option value, if any
                    !ISEG is the integer name of the
                    !detected segment, if any

C --- CHECK FOR "QUIT"

      IF (IPKID .EQ. 1) THEN  !quit was selected
         GO TO 80

      ELSE IF (IPKID.EQ.2) THEN !update was selected
         CALL JFRAME            !redraw all graphics
         GOTO 15

      END IF

C     --- DELETE THE SEGMENT FROM VIEW
                                !since no menu option was
                                !selected, the user must
                                !want to delete the segment
                                !ISEG from view

          CALL JVISBL (ISEG,0) !unpost the segment
          VISBL (ISEG) = .FALSE.  !record the deletion
```

```
            GO TO 10

80      CONTINUE

C --- ERASE THE PROMPT MESSAGE
      CALL JVISBL(PROMPT(4),0)

C --- MAKE THE DELETE MENU INVISIBLE
      CALL JVISBL (MENUS(9),0)

      END
```

The alternative to posting segments is to recreate the segments each time they are required. The alternative to unposting segments is to purge undesired segments from memory and redraw the graphics screen each time segments are deleted. Both of the these alternatives are very time consuming and add complexity to the implementation program. Posting and unposting is the better alternative for this application.

C. TRANSLATING AND SCALING SEGMENTS

Chart and Sketch provides the user the option of redrawing the graphics to use the entire graphics screen without displaying the menus and prompts. To implement this option, all the graphics in the drawing area of the screen must be translated (moved up and to the right) and scaled (ie: drawn larger). This would prove to be very slow in implementation if each individual segment were deleted, scaled, purged and recreated.

The usual procedure used by Chart and Sketch to move a segment consists of specifing a new relative position for the segment, deleting the segment, and recreating the segment in the new location. This proves an efficient method to move one segment, but becomes too slow if many segments are involved, eg: if every displayed segment is to be moved. In order to reduce the program response time (a design principle), a more global implementation is used.

DI-3000 provides a routine (JT2ALL) to translate and scale segments (in world coordinates) by changing the current transformation matrix (CTM). All graphics primitives (the collection of which was defined as a segment) are modified by the transformation matrix before being output to the graphics device. By modifing the CTM, all graphic primitives of a defined segment can be translated, rotated, and or scaled by a single operation.

In the following example from Chart and Sketch, the original transformation matrix is saved (using JVSAVE) and then each visible segment is translated, by changing the reference origin, and scaled so as to fill the entire graphics window (using JT2ALL). To return to the original display, the translated and scaled segments are purged from memory and deleted from the screen, the original transformation matrix is restored (using JVLOAD) and the original segments are posted (using JVISBL).

```
C --- VIEW THE MAP

      CALL JVSAVE (ARRAY)          !save the current
                                   !transformation matrix
      DO ISEG=101,NUMBER

        IF (VISBL(ISEG)) THEN      !if the segment is visible
          CALL JVISBL (ISEG,0)     !make it invisible

          CALL JT2ALL (ISEG, -1.,.15, 1.3,1.3, 0.,0.,.0)
                       !ISEG is the name of the segment
                       !the other parameters are the
                       !origin, scaling, rotation,
                       !and translation values

          CALL JSCOPY (ISEG,ISEG+1000,1)
                       !copy the original * transform

          CALL JVISBL (ISEG+1000,1)
                       !make transformed segment visible

        END IF
      END DO
      CALL JFRAME      !clean up the screen

C --- RETURN TO ORIGINAL DISPLAY WHEN USER PRESSES
C     A LOCATOR BUTTON

      TYPE *,'Press a LOCATOR button to continue.'
      CALL KLOCATD(1,1,10,IB,X,Y)
                  !wait for user to press button

      CALL JVLOAD (ARRAY)  !restore original transformation
                           !matrix

      DO ISEG=101,NUMBER
        IF (VISBL(ISEG)) THEN

          CALL JVISBL (ISEG+1000,0)
                           !delete transformed segments
          CALL JPURGE (ISEG+1000)
                           !purge transformed segments

          CALL JT2ALL (ISEG,.0,.0,1.,1.,0.,0.,0.)
                           !redefine transform matrix

          CALL JVISBL (ISEG,1)    !post original segments
        END IF
      END DO
```

D. ZOOMING AND PANNING

The zoom and pan function provided by Chart and Sketch utilizes two RAMTEK specific graphics processor codes. The first, OPCODE (26), replicates pixels by a power of two to sixteen by modifing the RM-9460 hardware zoom registers in the memory control processor (MCP). (The MCP draws the graphics primitives sent to it by the DI-3000 into the RAMTEK hardware refresh memory. The MCP controls the refresh address generation which in turn controls the RAMTEK hardware pan and zoom.) The second, OPCODE (27), redefines the hardware video origin of the MCP.

The ZOOM function in Chart and Sketch calls for a pixel replication factor of two (equivalent to magnifing the graphics on the screen by a power of two) using OPCODE (26). The PAN function makes use of the KECHO routine developed by CSC.

Using this routine (which is documented in appendix B to this thesis), OPCODE (26) is invoked to zoom and the drawing area of Chart and Sketch is mapped onto the graphics screen. This mapping is used to restrict the video origin to a value within the drawing area to prevent the user from panning into the menu or prompt areas (error accommodation). Panning is achieved by redefining the hardware video origin as the current cursor location through recurring calls of OPCODE (27) - immediate semeantic feedback.. An example of this procedure in Chart and Sketch is provided below.

In this example, the cursor location is defined to be the video origin by setting the cursor echo level to "10". The zoom value is set in the same routine to be equal to a magnification X2. The user is prompted to depress, and maintain depressed, a button on the locator. As long as the button remains depressed, the the cursor position will (re)define the video origin of the graphics screen. When the button on the locator is released the video origin remains defined as the last position of the cursor. The user is prompted to depress a button on the locator to return to the normal display. When a button is depressed, the video origin and zoom are reset to normal.

ZOOMING AND PANNING

```
C --- ZOOM AND PAN
18    CONTINUE

C     ---SET ECHO TO LEVEL 10.
C        Pan and Zoom using the RAMTEK hardware

      IARG(1) = 10     !Echo level 10 = zoom
      IARG(2) = 1      !Zoom value = 2X pixel replication

      CALL KTECHO(1,2,1,2,4,IARG,MAPV)  !Zoom and Pan
                                        !while button is
                                        !depressed
                                        !The origin is defined
                                        !as the current cursor
                                        !location.


      TYPE *,'Move the locator with a button depressed'
      TYPE *,'to PAN. Release the button when the desired'
      TYPE *,'view is on the screen.'

C --- GET LOCATOR INPUT FOR PAN

20    CALL JLOCAT (1,1,10,IB,X,Y) !get cursor coordinates
                                  !for video origin
```

```
      IF (IAND(IB,'20'X).EQ.0)GOTO 20
                                  !loop until button is
                                  !released

      TYPE *, 'Press a button on the locator'
      TYPE *, 'to return to normal view.'

C --- RESET THE PAN AND ZOOM TO NORMAL VIEWING PARAMETERS
25    CALL KLOCATD (1,1,1,IBUT,X,Y)
                                  !wait for button depression
         IF (IBUT.LE.0)GOTO 25    !check for error
      CALL JESCAP(9420,0,2,0,RARG)
                                  !reset video origin
      CALL JESCAP(9412, 1, 0, 0, RARG)
                                  !reset zoom to value 0
```

An alternate method to zoom would be to redefine a smaller world window, enable clipping, and mapping the window to a large viewport on the screen. The application program would have to define a geometric portion of the drawing area (a window) and then map this window onto the graphics screen: the smaller the defined window, the larger the effective zoom. Panning could be achieved in a similar manner, by changing the center of the defined window within the world (drawing) coordinates.

This method proves to be unsatisfactory in terms of program response time. It requires far too much execution time and is more difficult to implement. The KECHO routine, supplied by CSC, utilizes the speed of the RAMTEK hardware. Making use of the hardware zoom and pan is much.

E. FILING GRAPHICS

A key feature provided by Chart and Sketch is the ability to store graphic images in a directory file for later recall. A program which provides the ability to create graphic images, but which does not provide for the storage and later recall of those images is of very limited value. There are three classes of graphics storage: psuedo display files (or metafiles), storage display files, and implementation program files.

A pseudo display file is a graphics database, self-sufficient, and device-independent description of the graphics objects on the screen. It is often referred to as a metafile. Metafiles can be used to communicate graphics information between differing graphics hardware devices and application programs. Chart and Sketch does not utilize metafiles as it was developed at a facility not having a metafile generator/translator.

Storage display files (called section files in RAMTEK terminology) store the contents of the display refresh memory, pixel by pixel. This method provides rapid redisplay of graphics. The FILE option on the Main menu of Chart and Sketch provides the user with a way to save the graphics that have been created using Chart and Sketch by encoding the picture currently displayed on the graphics device and writing it to a file in any system directory.

Section files are created and read using four CSC graphics routines and are therefore interprogram portable. (Any program written using DI-3000 graphics language can read a section file created by Chart and Sketch.) However, section files do not store the graphics image as logical contructs, or segments; therefore, a segmented graphics image recreated from a section file cannot be modified. Section files also use large amounts of storage since (as implemented in Chart and Sketch) the entire graphic's screen bit planes must always be encoded.

The section file is created and mapped into the user's virtual address space using KSCRET (to create and map section file) or KSOPEN (to open and map section file). Escape code 9425 encodes the RAMTEK screen (into pixel data) and creates the section file.

Escape code 9424 reads the section file and writes the graphics on all RAMTEK RM-9460's initialized by Chart and Sketch. It provides the fastest option available in Chart and Sketch to transfer image data from a file to the RAMTEK screen.

The two following examples from Chart and Sketch illustrate the use of the CSC routines to create and then read a section file. This equates to effectively storing and then retrieving a graphics display.

In the first example, a window is defined which includes the entire graphics screen. KSBYTE24 is called to determine the storage area needed for the graphics file. A section file is created in virtual memory by calling KSCRET and the pixel information is coded and written to the section file by calling the JESCAPE 9425 routine. The program then copies the section file in virtual memory to the user's directory, deletes the virtual section file, and deassigns the section. This procedure is reversed to read a section file.

GRAPHICS STORAGE USING A STORAGE DISPLAY FILE

```
C --- SET THE VIRTUAL wind (or viewport)
C     FOR READING TO FILE

          CALL JASPEK (1,RATIO) !this will write
          wind(1) = -1.         !the entire graphics
          wind(2) = -RATIO      !screen
          wind(3) =  1.
          wind(4) =  RATIO

C --- GET THE PIXEL OR BYTE COUNT

      NBYTE = KSBYTE24(1,wind(1),wind(2),wind(3),wind(4))

C --- CREATE THE SECTION FILE (name = FILENAME, size = NBYTE,
C     channel = T_CHAN, address = ADDR(1) to ADDR(2))

      CALL KSCRET(FILENAME, NBYTE, LUN, T_CHAN, ADDR, IERR)
      IF (IERR .NE. 0) THEN
        CALL CLRSCN
        TYPE *, 'Error in creating the file ',FILENAME
        GO TO 90
      ENDIF

C --- WRITE THE PIXEL DATA TO THE SECTION FILE

      TYPE *,'Writing graphics to file.  Please stand by.'
      CALL JESCAP(9425,1,4,ADDR,wind)

C --- UPDATE THE SECTION FILE. Copy the section file in
C     virtual memory to the corresponding section disk file.
```

```
      CALL LIB$GET_EF(IEF)
      IS = SYS$UPDSEC(ADDR, , , , %VAL(IEF), , , )
      ISS = SYS$WAITFR(%VAL(IEF))
      CALL LIB$FREE_EF(IEF)
      TYPE *,FILENAME, ' has been written.'

C --- DELETE THE SECTION FILE FROM VIRTUAL ADDRESS SPACE
      IS = SYS$DELTVA(ADDR,,)
      IF (.NOT.IS) THEN
         CALL CLRSCN
         TYPE *, ' Error in removing the section.'
      ENDIF

C --- DEASSIGN THE SECTION
      IS = SYS$DASSGN(%VAL(T_CHAN))
      IF (.NOT.IS) THEN
        CALL CLRSCN
        TYPE *,'Error in deassigning the channel.'
      ENDIF

C --- CLOSE THE FILE
      CLOSE (UNIT=LUN)

90    CONTINUE

C --- PROMPT THE USER TO ENTER INPUT
C --- AT THE GRAPHICS TERMINAL

       TYPE *,'Enter selection from the GRAPHICS DEVICE.'
       END



          READING GRAPHICS USING A STORAGE DISPLAY

C --- READ A SECTION FILE FROM THE DIRECTORY AND WRITE IT
C     ONTO THE SCREEN

C --- OPEN THE SECTION AND GET THE VIRTUAL ADDRESS (= PADDR)
C     (KSOPEN is a CSC  DI-3000 routine)

      CALL KSOPEN(FILENAME, LUN, ICHAN, PADDR, IERR)

C --- OPEN THE SECTION FILE 'FILENAME' ON CHANNEL 'ICHAN'
C     AS VIRTUAL ADDRESS 'PADDR'
      IF (IERR .NE. 0) THEN
          CALL CLRSCN
          TYPE *,'Error in opening file. RETRIEVE ABORTED.'
          GOTO 90
      ENDIF
```

```
C --- SET THE WINDOW FOR READING THE FILE ONTO THE SCREEN
C     USE THE ENTIRE SCREEN
          CALL JASPEK (1,RATIO)  !this will use the
          wind(1) = -1.          !entire screen
          wind(2) = -RATIO
          wind(3) =  1.
          wind(4) =  RATIO

C --- READ THE FILE USING ANOTHER CSC ROUTINE
      CALL JESCAP(9424, 1, 2, PADDR, wind)

C _____________________________________________________________

C  note:  This portion of the routine is VMS-specific.
C  It has nothing to do with DI-3000

C _____________________________________________________________

C --- ERASE THE MAP FROM VIRTUAL ADDRESS SPACE
      IS = SYS$DELTVA(PADDR,,)
      IF (.NOT.IS) THEN
         CALL CLRSCN
         TYPE *, 'Error in removing the section.'
      ENDIF

C --- DEASSIGN THE SECTION
      IS = SYS$DASSGN(%VAL(ICHAN))
      IF (.NOT.IS) THEN
        CALL CLRSCN
        TYPE *,'Error in deassigning the channel.'
      ENDIF

C --- CLOSE THE FILE
      CLOSE (UNIT=LUN)

90    CONTINUE

C --- PROMPT USER TO RETURN TO THE GRAPHICS DEVICE FOR INPUT
      TYPE *,'Enter selection from the GRAPHICS DEVICE.'
```

Implementation program files store the logical constructs on the screen in a manner unique to the program. The filetype of "Map" is used to store charts; the filetype of "data" is used to store all other graphics. Two different filetypes are used to prevent the user from

accidentally deleting a data or map file. (Eg: Suppose the user created and filed a chart called "India", and then created and filed a graphics display called "India"; Another version of the "India" file would be created. The first time the user's directory was purged, the original "India" chart would be deleted.)

The data and map files of Chart and Sketch use minimal storage space and can be modified after being stored and recalled. The data and map files take longer to redisplay then section files if the display contains many or complex segments. They cannot be used by any program (other than Chart and Sketch) to recreate the filed graphics. They are not interprogram portable. The following examples from Chart and Sketch illustrate the use of a implementation program file.

Before the file can be created, the program must encode the parameters necessary to recreate the display. Chart and Sketch uses separate arrays for each parameter and indexes the parameter arrays by the number of the segment which the parameters represent.

After storing all segment parameters, the program opens a directory file. The parameters of each visible segment are then written to the file using FORTRAN-77 default formatting.

To read a program file: Chart and Sketch opens the file, reads the parameters of a segment using the FORTRAN-77 default formatting, and then calls subroutine Sketch to create a segment using the parameters read from the file. The program loop repeats this process until the end of the file is reached.

GRAPHICS STORAGE USING A PROGRAM FILE

```
C --- STORE SEGMENT USING PROGRAM PARAMETERS
      DO I = 1,100                !store up to 100
         XARRAY(SEGMNT,I)=X(I)    !coordinates that
         YARRAY(SEGMNT,I)=Y(I)    !define the segment
      END DO

      DO I=1,80
         LTRS(SEGMNT,I)=LTR(I)    !store up to 80 text
      END DO                      !TEXT characters
      FILLS(SEGMNT) =FILL         !is the segment filled ?
      INDEXS(SEGMNT)=INDEX        !what segment type ?
      COLORS(SEGMNT)=COLOR        !what is it's color ?
      SIZES(SEGMNT) =SIZE         !if text, how large ?
      FONTS(SEGMNT) =IFONT        !if text, which font ?
      PNTS(SEGMNT)  =POINTS       !how many points defined ?


C --- ENCODE THE GRAPHIC SEGMENTS DISPLAYED AND WRITE
C     THEM TO A FILE OF filename AND TYPE dat

       OPEN (UNIT=4,NAME=FILENAME,TYPE='NEW',ERR=20)
       DO I=BEGIN,NUMBER          !for all visible segments
          IF (VISBL(I)) THEN      !write their program
             FILL  =FILLS(I)      !parameters to a file
             POINTS=PNTS(I)
             INDEX=INDEXS(I)
             COLOR=COLORS(I)
             SIZE=SIZES(I)
             IFONT=FONTS(I)
             DO J = 1,100
                X(J)=XARRAY(I,J)
                Y(J)=YARRAY(I,J)
             END DO
             DO J = 1,80
                LTR(J)=LTRS(I,J)
             END DO
       WRITE(4,*,END=15,ERR=25)
      *POINTS,X,Y,INDEX,COLOR,FILL,SIZE,IFONT,LTR
```

```
         END IF
      END DO
15    CONTINUE
      TYPE *,'File ',FILENAME,' has been written.'
      TYPE *,'Enter selection at the GRAPHICS TERMINAL.'
      CLOSE (UNIT=4)
      RETURN
20    CONTINUE
      CALL CLRSCN
      TYPE *,'Error in openning file ',FILENAME,'.'
      TYPE *,'Enter selection at the GRAPHICS TERMINAL.'
      CLOSE (UNIT=4)
      RETURN
25    CONTINUE
      CALL CLRSCN
      TYPE *,'Error in writing file ',FILENAME,'.'
      TYPE *,'Enter selection at the GRAPHICS TERMINAL.'
      CLOSE (UNIT=4)
      END
```

READING GRAPHICS USING A PROGRAM FILE

```
C---  READ A PROGRAM FILE OF SEGMENT PARAMETERS
C     UNITL END OF FILE

      OPEN (UNIT=4,NAME=FILENAME,TYPE='OLD',ERR=20)

10    READ(4,*,END=15,ERR=20)
     *     POINTS,X,Y,INDEX,COLOR,FILL,SIZE,IFONT,LTR


C---  PASS THE SEGMENT PARAMETERS TO SKETCH TO GENERATE
C     EACH SEGMENT IN THE FILE

      CALL SKETCH (POINTS,X,Y,INDEX,COLOR,SIZE,
     *             IFONT,LTR,FILL)
      GOTO 10

15    CONTINUE
      CALL CLRSCN
      TYPE *,'File ',FILENAME,' has been read.'
      TYPE *,'Enter selection at the GRAPHICS TERMINAL.'
      CLOSE (UNIT=4)

      RETURN
20    CONTINUE
      CALL CLRSCN
      TYPE *,'Error in reading file ',FILENAME,'.'
```

```
TYPE *, 'Enter selection at the GRAPHICS TERMINAL. '
CLOSE (UNIT=4)
END
```

Chart and Sketch also allows the user to scale and translate the display of data and map files. This allows the original graphics display to be reduced in size and located anywhere on the graphics screen independent of its original size and location. This was implemented by scaling and translating the segment parameters contained in the data and map files.

In the following example from Chart and Sketch, the user designates the window (into which the original graphics display is to be mapped and drawn) as a rectangle on the graphics screen. The vertical aspect of the rectangle is adjusted by Chart and Sketch to prevent distortion of the display and the segment parameters are scaled and translated.

The program describes the scaling and translating option to the user on the alphanumeric terminal. It then prompts the user to select "YES" from the menu on the graphics screen to use the option. If "NO" is selected, the program procedes as described above. If "YES" is selected the user is prompted to enter two points defining a rectangle on the graphics screen.

The vertical aspect of this rectangle is adjusted such that the horizonta/vertical ratio of the rectangle equals that of the drawing area (to prevent distortion of the

display). The parameters describing a segment are read from the file and scaled to fit within the rectangle. Finally, subroutine Sketch is called to create the segment using the scaled parameters.

SCALING AND TRANSLATING PROGRAM FILES

```
      OPEN (UNIT=4,NAME=FILENAME,TYPE='OLD',ERR=20)

      TYPE *,'When recalling a data file the user has'
      TYPE *,'the option of redrawing the graphics to'
      TYPE *,'their original size (which will use the'
      TYPE *,'FULL VIEWing SPACE), or to a reduced '
      TYPE *,'size (which will use only a selected '
      TYPE *,'portion of the viewing space.'
      TYPE *,' '
      TYPE *,'DO YOU WANT TO USE THE FULL VIEWING SPACE ?'
      TYPE *,'(Enter YES or NO from the graphics device.)'


81    CALL KLOCATD(1,1,1,IB,X1,Y1)    !get locator value
      IF(IB.LT.0)GOTO 81
      CALL JSPICK (X1,Y1,1,ISEG,IPIC)!correlate coordinates
                                      !with a menu option
                                      !Y = yes, N = no

        IF (IPIC.EQ.2)THEN
           ANSWER = 'N'
        ELSE IF (IPIC.EQ.1) THEN
           ANSWER = 'Y'
        ELSE IF (ISEG.LT.0) THEN
           GOTO 81
        END IF

      IF (ANSWER.EQ.'Y')THEN      !if YES then
         XTRANS=1.                !don't scale and
         YTRANS=1.                !don't translate
         GOTO 8

      ELSE IF (ANSWER.EQ.'N') THEN    !if NO then
                                      !then get retangle

C --- PROMPT USER FOR FIRST POINT OF RECTANGLE
         CALL JVISBL (PROMPT(5),1)

C --- READ POINT COORDINATES FROM LOCATOR
2        CALL KLOCATD(1,1,1,IB,WINDOW(1),WINDOW(2))
         IF (IB .LE. 0) GO TO 2
```

```
C --- PROMPT USER FOR DIAGONALLY OPPOSITE POINT
         CALL JVISBL (PROMPT(5),0)
         CALL JVISBL (PROMPT(6),1)

C --- READ POINT COORDINATES FROM LOCATOR
3        CALL KLOCATD(1,1,1,IB,WINDOW(3),WINDOW(4))
         IF (IB .LE. 0) GO TO 3

C --- CALCULATE THE RECTAGLE CORNER COORDINATES

          WIND(1) = AMIN1(WINDOW(1),WINDOW(3))
          WIND(4) = AMIN1(WINDOW(2),WINDOW(4))
          WIND(3) = AMAX1(WINDOW(1),WINDOW(3))
          WIND(2) = AMAX1(WINDOW(2),WINDOW(4))

          DO I=1,4
             WINDOW(I) = WIND(I)
          ENDDO

C --- MAP THE INPUT (LOCATOR) SCREEN COORDINATES TO THOSE
C     USED TO DEFINE THE DRAWING AREA (WORLD COORDINATES)
C     --- THIS DOES THE TRANSLATION

      CALL JCONVW(WINDOW(1),WINDOW(4),XDISP,YDISP,Z)

C --- CALCULATE THE SCALING NEEDED

      XTRANS=ABS((WINDOW(1)-WINDOW(3))/(MAPV(3)-MAPV(1)))
      YTRANS=XTRANS*(MAPV(4)-MAPV(2))/(MAPV(3)-MAPV(1))

      END IF

8     CONTINUE

C --- READ THE PROGRAM FILE

10    READ(4,*,END=15,ERR=20)
     *      POINTS,X,Y,INDEX,COLOR,FILL,SIZE,IFONT,LTR

C --- SCALE THE SEGMENT PARAMETERS

       DO I =1,POINTS
          X(I)=(X(I)*XTRANS)
          Y(I)=(Y(I)*YTRANS)
       END DO
       SIZE=SIZE*XTRANS

C ---  CREATE THE SEGMENT USING THE SCALED PARAMETERS

       CALL SKETCH (POINTS,X,Y,INDEX,COLOR,SIZE,IFONT,LTR,FILL)
       GOTO 10
```

Since storage display and implementation program files offer such differing capabilites, Chart and Sketch incorporates both and allows the user to determine which method of storing graphics is best suited to his particular application. The following table summerizes the differences between data, map, and section files as implemented in Chart and Sketch.

FILE TYPE	DATA	MAP	SECTION
Retrieval speed	slow	slow	fast
Storage size	small	small	large
Changeable ?	yes	yes	no
Portable ?	no	no	yes
Created by menu	MAIN	CHART	MAIN
Filetype	.DAT	.MAP	.SEC

Table 1. Chart and Sketch Graphics Files

IV. USE OF CHART AND SKETCH

The purpose of Chart and Sketch is to allow a novice graphics user to quickly and effectively utilize the graphics capabilites of the DI-3000 software and the RAMTEK RM-9460 Graphic Display System without any knowlege of FORTRAN or DI-3000 programming.

A. IN THE COMMAND AND CONTROL ENVIRONMENT

In addition to and concurrent with the primary objective, Chart and Sketch may find application in sundry Command and Control environments where the graphic display of information, data, charts, and symbology simplifies (or could simplify) situation assessment, history, or tracking.

Furthermore, Chart and Sketch could find ready application as a means of communicating graphic information over standard data or voice lines wherever file transfer capability and protocol exist. Section files could be transmitted, received, and then used to reconstruct complex graphic information using the CSC implementation of DI-3000 at both the transmitting and receiving sites (presupposing compatible operating systems and/or modification to the application program). If both sites had access to Chart and Sketch, graphic information could be transmitted in a

modifiable, updateable form. This implies the possiblity of a new form, a GRAPHIC form, of unit-to-headquarters and unit-to-unit reporting.

When combined with available hardware, Chart and Sketch also provides a ready, easy facility for producing professional quality typesetting, overhead transparencies, and slides.

The following example provides an illustration of the application of Chart and Sketch to a Command and Control environment. In this example, charts and symbols are employed to represent/track unit location, type and status. Unit location is represented by its physical location on the chart. Unit type is represented by a unique symbol (icon), and unit status is represented by the color of the icon.

Given the above assumptions, the following illustrates the steps which might be used to implement Chart and Sketch as an effective Coammand and Control tool. (This is intended as an example. A more detailed explanation of the procedures involved can be found in Appendix A to this thesis.)

(1) Use Chart and Sketch to create a CHART database. Chart and Sketch provides the facility for creating a chart database by tracing the hardcopy of a map or chart with the locator/pick devices. Chart and Sketch then creates a permanent file (eg: India.map) containing the data necessary to redraw the chart on the graphics screen at any

time. (For the details involved, the reader is referred to Appendix A to this thesis, section 3.3.2.) This step may not be needed if a database already exists and Chart and Sketch is modified to read the existing database.

(2) Use Chart and Sketch to create an ICON database. Chart and Sketch can also be used to create an ICON database. The user would draw the ICON using an option from the Sketch menu (ie: rectangle, diamond, circle, sector or polygon). The color is not important, but the interior attribute should be "filled". When the ICON is complete, the user would file the ICON by selecting the FILE option from the main menu, selecting a file name for the ICON (eg: tank), and selecting a DATA type file. Chart and Sketch will then create a file called TANK.DAT which could be used as a database to draw tanks. This procedure would be repeated for each ICON. (A more detailed explanation is available in Appendix A to this thesis, sections 3.4 and 3.5.)

(3) Use Chart and Sketch to display CHARTS and ICONS. Assume that proper command and control requires a display which shows the current location, type and status of multiple units in India. First, the user would draw a map of India on the graphics screen by using the Chart menu. Next the user would draw one of each desired ICON on the graphics screen using the RETRIEVE option from the Main menu.

To position an ICON on the chart, and at the same time determine its color (representing its status), the user would select a color using the COLOR option from the Sketch menu and then make a copy of the ICON in the desired position using the COPY option. Chart and Sketch would create a duplicate of the ICON at the desired position using the selected color. The same procedure would be used to place any number of ICONS of the same or different color in selected positions. When all ICONS were positioned, the user would delete the original or undesired ICONS using the DELETE option from the Sketch menu.

(4) Use Chart and Sketch to update the ICONS. Chart and Sketch provides the facility to move an ICON from one location on the chart to another. Suppose that a recent report indicates that tank #5 has moved fifty miles north. The user would use the MOVE option from the Main menu to move the ICON representing tank #5 fifty miles north on the chart. With one operaton, the chart would be current. Nothing else on the chart would need to be moved or changed.

Now suppose that contact is lost with tank 5 (a red status). To change the ICON representing tank 5 from yellow to red, the user would use the COPY option from the Sketch menu and make a duplicate of the ICON in the same location but using the red color characteristic. The user would then delete the yellow ICON using the DELETE option.

Chart and Sketch would provide a simple and fast method for maintaining the display (used for command and control) current.

(5) Use Chart and Sketch to file and retrieve Command and Control Displays. Once a display is created, it can be filed in the user's directory. This display can then be retrieved at a later date for redisplay and/or modification. This would be useful for maintaining multiple Command and Control Displays and for presenting multiple displays quickly during briefings.

Other uses for Chart and Sketch, (eg: creating and presenting slides) could be developed according to specific needs.

B. AS A GENERAL APPLICATION GRAPHICS TOOL

Chart and Sketch provides the user a ready demonstration, as well as the opportunity to experiment and become familiar with, the basic capabilites of interactive, computer, color graphics. The user with only a rudimentary knowledge of FORTRAN and DI-3000 can study and refer to the construction of Chart and Sketch in the design of interactive graphics programs implementing any or all of the basic graphics capabilites employed by Chart and Sketch. For the user more experienced in programming, it provides an opportunity to evaluate, criticize and improve the techniques of interaction, graphics display and storage.

V. RECOMMENDED ENHANCEMENTS

This section will address some of the possible enhancements which would make Chart and Sketch a more efficient and user friendly graphics tool

A. WORLDWIDE CHART DATABASE ACCESS

As previously stated, an important, forseeable use of Chart and Sketch includes the Command and control environment where the display of data, charts, and symbology could simplify situation assessment. As currently written and implemented, Chart and Sketch does not have access to a worldwide database to draw charts. The program does provide a facility for inputing charts manually, but the method is slow and of limited value. Access to a worldwide data base is need for drawing charts, such as that employed by the Navy's Interim Battle Group Tactical Trainer (IBGTT). [Ref. 8].

Chart and Sketch was designed to provide for this additional facility. All the procedures involved in the creation or reading of charts are invoked from the Chart Menu. The program functions that are unique to the Chart Menu are provided by the subroutine Wdata. No other program functions or subroutines are dependent on the functions

provided by the Chart Menu or Wdata subroutine. Only the Chart Menu and Wdata subroutines would have to be modified to add a routine to access a worldwide chart database. The routine could be designed to draw a chart given any latitude, longitude, and radius using the Polyline routine in the Sketch subroutine.

The access to a worldwide chart data base would enhance the spectrum of conceivable applications and greatly facilitate the use of Chart and Sketch as a general use graphics tool.

B. MULTIPLE SEGMENT OBJECTS

Chart and Sketch defines each graphics segment (collection of graphics commands used in creating a graphics object) by an integer name. This integer name is then used by the program to identify the segment for deletion, duplication or movement. As currently implemented, a new segment is defined each time the user selects an option from the Sketch menu.

If the user selects CIRCLE from the Sketch menu and draws circle, and then selects polygon and draws a polygon within the circle: two separate segments, each with its own integer name are created.

Chart and Sketch does not currently provide any way to define both the circle AND the polygon as one segment. To delete, duplicate or move the circle-polygon combination,

the user must delete, duplicate or move the circle and polygon individually.

The ability to redefine multiple segments as one segment would greatly enhance the user friendliness of Chart and Sketch. This capability could be implemented by allowing the user to first select a BEGIN SEGMENT option from the Sketch menu, draw multiple segments, and then select an END SEGMENT option. The program could then redefine all those segments created between the BEGIN and END segment options as one logical segment. This would permit the user to move, duplicate and delete the object as one segment rather than many.

The method of storing segment attributes and creating/reading program graphics files would need to be modified if multiple segments were defined as one. This could be accomplished in at least two ways. First, an END OF SEGMENT flag could be added to the attributes that define each segment. The delete, duplicate and move subroutines would need to be modified to delete, duplicate or move all segments from the selected segment until an END OF SEGMENT flag was encountered. However, the user would need to select the lowest numbered REAL segment composing the selected LOGICAL segment or the program would have to index each REAL segment to its LOGICAL segment.

A second, and perhaps better, method of implementing multiple segment definition would involve adding an index into the arrays which store segment attributes. This index would define which REAL segments compose one LOGICAL segment. Using this method most of Chart and Sketch could stand as written. The delete, duplicate and move routines would have to be modified to delete, duplicate or move all the REAL segments that compose the selected LOGICAL segment.

C. COLOR AND FILL

The current implementation of DI-3000 initializes the RAMTEK RM9460 in such a way that some undesirable features result. If graphics segments are drawn such that they intersect on the graphics monitor, the RAMTEK RM-9460 may or may not mix the two colors at the intersecting points. This feature of the DI-3000 implementation of the RAMTEK is unavoidable in the current configuration.

This feature has very undesirable results when using Chart and Sketch. An escape feature needs to be added to the DI-3000 library to prevent the RM-9460 from mixing segment colors.

Another problem is associated with intersecting segments. The RAMTEK RM-9460 fills segments starting from the center. If a "filled" graphics segment is drawn intersecting another segment, the latter segment will only be "filled" in the none-intersecting region that contains

the center of the graphics segment. The segment will only be filled to the edge of the intersecting segment. This is also a feature of the DI-3000 implementation of the RAMTEK hardware. Another DI-3000 escape routine needs to be designed to overcome this problem as well.

Preventing the mixing of colors and allowinf the filling of entire segments when they intersect would greatly enhance the useability of Chart and Sketch.

APPENDIX A

CHART AND SKETCH USER'S MANUAL

1. INTRODUCTION 78

2. PROGRAM CONCEPT 80

3. USER INSTRUCTIONS 81

3.1 LOGIN PROCEDURES 81

3.2 MAIN MENU 82

3.3 CHART MENU 85

3.3.1 Read a Chart File 86

3.3.2 Make a Chart File 88

3.3.3 Examples 91

3.3.4 Help 91

3.3.5 Quit 91

3.4 SKETCH MENU 92

3.4.1 Rectangles 93

3.4.2 Diamonds 94

3.4.3 Circles 95

3.4.4 Polygons 96

3.4.5 Lines 98

3.4.6 Arcs and Sectors 99

3.4.7 Text 101

3.4.8 Colors 102

3.4.9 Fill Option 104
3.4.10 Deletions 105
3.4.11 Copies 107
3.4.12 Quiting the Sketch Menu 108
3.5 CREATING FILES 109
3.6 RETRIEVING FILES 113
3.7 CLEARING THE SCREEN 117
3.8 VIEWING THE MAP 118
3.9 MOVING SEGMENTS 118
3.10 ZOOMING AND PANNING 119
3.11 QUITING 120
4. ADDITIONAL INFORMATION 122

LIST OF FIGURES

Figure A-1: Main Menu..... 83
Figure A-2: Chart Menu..... 86
Figure A-3: Quit Menu..... 89
Figure A-4: Sketch Menu..... 93
Figure A-5: Text Menu.....101
Figure A-6: Color Menu.....103
Figure A-7: Update Menu.....106
Figure A-8: Verify Menu.....117

LIST OF TABLES

Table 1: Feature Table 99
Table 2: Chart and Sketch Filetypes 110

LIST OF EXAMPLES

Example 1: Login and Initialization 82
Example 2: Read a Chart File - no errors 88
Example 3: Read a Chart File - with errors 88
Example 4: Creating a Section File - no errors 110
Example 5: Creating a Section File - with errors 111
Example 6: Creating a Data File - no errors 112
Example 7: Retrieving a Section File - no errors 114
Example 8: Retrieving a Data File - no errors 115
Example 9: Retrieving a Data File - with errors 116

1. INTRODUCTION

Chart and Sketch is a highly interactive, color, computer graphics program. The purpose of Chart and Sketch is to allow a novice graphics user to effectively utilize the graphics capabilites of the DI-3000 graphics software and the RAMTEK RM-9460 Graphic System without any knowlege of FORTRAN or DI-3000 programming.

Chart and Sketch provides for the interactive creation of multi-colored, empty and color-filled graphic images, including circles, diamonds, squares, polygons, lines, arcs, secotrs, charts and text. Furthermore, the program provides the facility to interactively move, duplicate, and delete the the graphic images and text strings as well as the capability to zoom and pan. The results of a graphics session (the image on the graphics monitor), may be written to file and recalled for display at a later date.

The program interaction is designed around menu selection using a RAMTEK color monitor, a graphics tablet and occassional keyboard entry from a terminal. The program menus and options are discussed in detail in this manual.

Chart and Sketch is resident on the Secure Wargaming Analysis and Research Laboratory (WAR LAB) at the Naval Postgraduate School (NPS) , Monterey, CA.

2. PROGRAM CONCEPT

Chart and Sketch is a menu driven, highly interactive, color graphics program. Each of the menus is presented in a logical, hierarchial process which makes the creation of graphics segments, (the image created by a collection of graphics commands) as simple and straightforward as possible.

Chart and Sketch creates graphics segments for the user automatically. These segments can be moved, copied, or deleted by the user without any knowledge of the process involved. Chart and Sketch employs simple user prompts which help the user to effectively utilize the program with a bare minimum of training.

The use of Chart and Sketch is discussed in detail in this manual.

3. USER INSTRUCTIONS

This section describes the use of Chart and Sketch in detail. Initiating the program and use of the hardware are described. Each program menu and its options are discussed in detail and program limitations are noted.

3.1 LOGIN PROCEDURES

In order to run Chart and Sketch, the user must have available a minimum of one VT100/102 terminal and one pair of RAMTEK graphics monitors with an associated locator and graphics pad. The program will run on multiple pairs of RAMTEK monitors if they are available and the user so chooses.

To run Chart and Sketch the user must access the program by giving the system command SKETCH. The program will then run in the user's directory. The program will ask the user to select which set of graphics monitors are to be utilized.

The War Lab currently supports multiple graphics monitors. Each monitor is assigned to a monitor set. When a monitor set is selected each monitor belonging to that set will display the graphics created by Chart and Sketch. From one to three monitor SETS may be selected. The monitor set to which a monitor is assigned is a number from one to three located beneath the monitor screen. (Note the use of "?" to

invoke the program help function. The help function returns more specific information about the question asked.)

Example. Login and Initialization

SKETCH

Enter the number of monitor SETS to use. (1-3 or ?)

?

The War Lab currently supports multiple graphics monitors. Each monitor is assigned to a monitor set. When a monitor set is selected each monitor belonging to that set will display the graphics created by Chart and Sketch. From one to three monitor SETS may be selected.

How many monitor SETS do you want to use?

1

Enter a monitor SET number. (1-3 or ?)

3

The program is loading. Please stand by.

3.2 MAIN MENU

The main menu is the first menu presented to the user after selecting the graphics devices to be utilized. This section will briefly describe the MAIN MENU options which will be described in greater detail later.

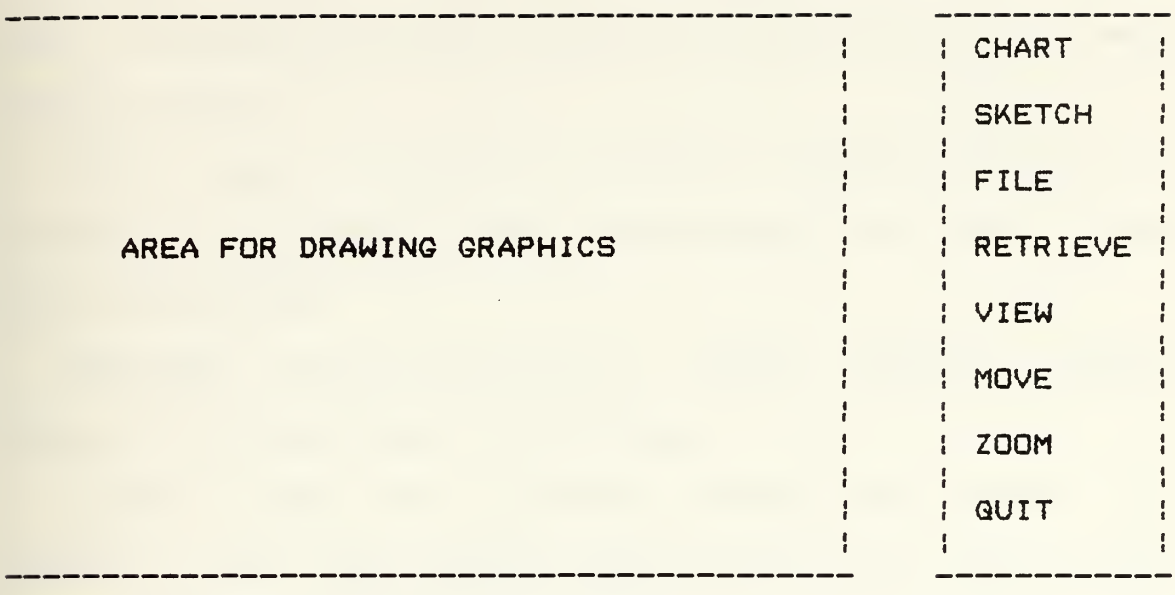

Figure A-1. Main Menu

A menu option is selected by moving the LOCATOR on the graphics tablet until the cursor is over the center of the desired option and then pressing any button on the locator. (Chart and Sketch does not distinguish between buttons on multi-button locators.)

The program is exited by selecting the option QUIT from this menu. The options which the user may select from this menu include:

CHART. Draw a selected chart from the database. If CHART is selected, the user will be presented another menu from which to select a more specific action.

SKETCH. Draw, delete, or copy graphics segments as selected by the user, including circles, diamonds, rectangles, polygons, lines, arcs, and text. Sketch also allows the user to select the color and fill characteristics

of the selected segments. If SKETCH is selected, the user will be presented another menu from which to select a more specific action.

FILE. Save a currently displayed graphics product by encoding the graphics data and writing it to a file in the user's directory.

RETRIEVE. Draw a previously created graphics product from a file in the user's directory.

CLEAR. Clear the graphics screen and delete all previous work done during the session which has not been saved to a file.

VIEW. Redraw the currently displayed graphics product to fill the entire monitor screen, (do not display prompts or menus). If VIEW is selected, the action will be executed without further user interaction. User directions (prompts) will appear on the VT100/102 terminal screen to allow the RAMTEK graphics screen to display only the graphics product.

MOVE. Move a currently displayed graphics segment to another location on the monitor screen. If MOVE is selected, the main menu will disappear and a prompt for appropriate user action will appear.

ZOOM. Magnify the currently displayed graphics product by a factor of 2 and allow the user to interactively change the reference center of the display (pan). If ZOOM is selected, the main menu will disappear and a prompt for appropriate user action will appear on the VT100/102

terminal screen to allow the RAMTEK graphics screen to display only the graphics product.

QUIT. Exit the program. If QUIT is selected, the program will end after the user has verified his intention to exit the program.

PROGRAM LIMITATION. No more than 1000 segments may be visible on the graphics terminal at any one time. A warning message will appear at the VT100/102 terminal, and a warning bell will ring at the VT100/102 terminal, when less than five of the 1000 segments are left. The user must then either limit the number of additional segments or delete previous segments.

The menus in Chart and Sketch are presented in a heirarchial structure. Each successive level requests more specific information about the desired graphics action. The user may return to a higher-level menu by selecting QUIT. If QUIT is selected from the MAIN MENU, the program will end. More specific information on each menu option is presented below.

3.3 CHART MENU

If CHART is selected from the main menu, the user will be presented with the CHART MENU and the prompt SELECT FROM MENU. Selections from this menu either identify a previously filed chart, to be drawn on the graphics monitor, or a new chart, to be created using the graphics devices.

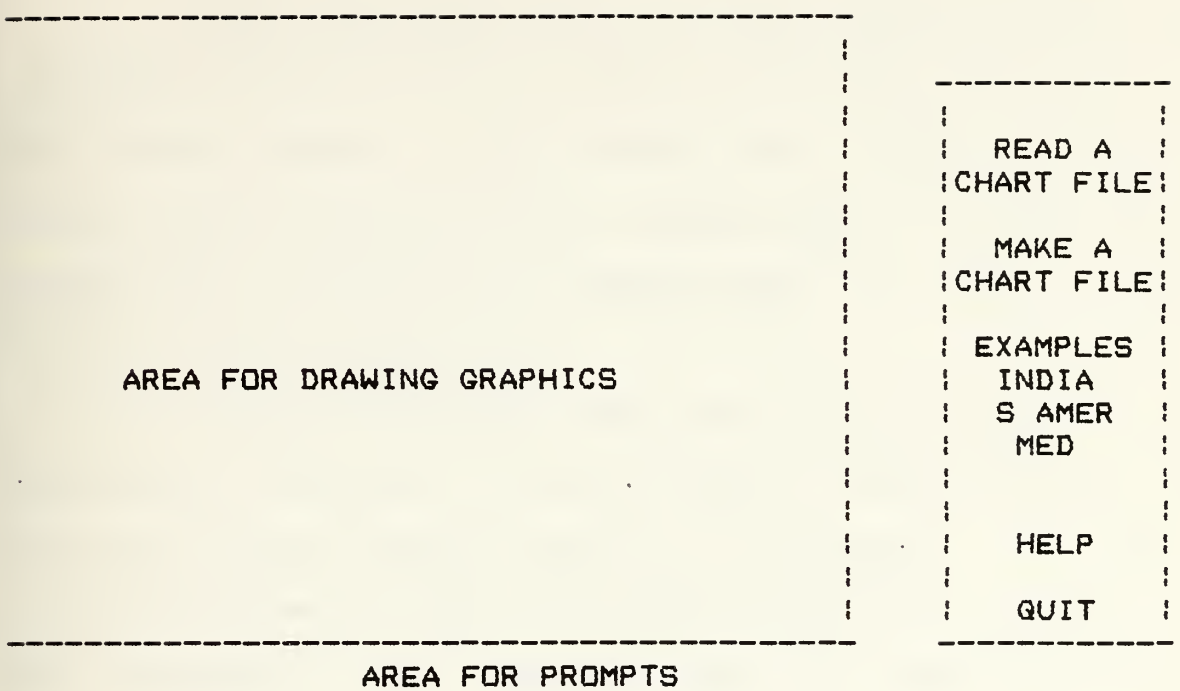

Figure A-2. Chart Menu

3.3.1 Read a Chart File

This option on the CHART menu provides the user with a way to display charts that were previously created using the MAKE A CHART FILE option.

The program will prompt the user (on the VT100/102) to enter the filename of the chart to be displayed on the graphics device. The user must enter the filename (from the VT100/102). At this time two things will happen, (1) the verify menu with YES and NO options will appear (refer to figure A-8) with the prompt, USE FULL VIEWSPACE ? ; and, (2) an explanation of the prompt will appear on the VT100/102 terminal.

When recalling a chart the user has the option of redrawing the chart to its original size (which will use the FULL VIEWing SPACE), or to a reduced size (which will use only a selected portion of the viewing space. The user must select YES, to use the full viewing space, or NO, to use only a selected portion of it.

If NO is selected, the user will be prompted, at the graphics terminal, to use the graphics tablet locator to enter the desired display area as a rectangle. The user will be prompted to enter two diagonally opposite corners of the rectangle. The vertical aspect of the specified rectangle will be adjusted such that the rectangle in which the chart is drawn has the same horizontal/vertical ratio as the drawing area, to prevent a distortion of the chart. The chart will be drawn within the adjusted rectangle. If YES is selected from the verify menu, the chart will be drawn using the entire drawing area. After the chart has been drawn, the user is returned to the main menu and prompted to make another selection. (See the example below.)

If an error is encountered in finding or reading the file, the user will be given a short error message and returned to the main MENU with the prompt to select from the menu. An example is shown below. The program prompts are indented.

Example 2. Read a Chart File - no errors

```
    Enter the file name.

DI3000

    When recalling a data file the user has
    the option of redrawing the graphics to
    their original size (which will use the
    FULL VIEWing SPACE), or to a reduced
    size (which will use only a selected
    portion of the viewing space.
    DO YOU WANT TO USE THE FULL VIEWSPACE?
    (Enter YES or NO from the graphics terminal.)

NO  --- from the graphics terminal

    (THE USER IS PROMPTED TO ENTER TWO POINTS
    DEFINING A RECTANGLE USING THE LOCATOR)

--- the user enters two points at the graphics terminal

    (THE FILE IS DISPLAYED WITHIN THE RECTANGLE
    AND THE MAIN MENU IS PRESENTED)
```

Example 3. Read a Chart File - with errors

```
    Enter file name.

DI3010      --- suppose that DI3010 does not exist ---

   ERROR in reading file     DI3010.MAP .
   Enter selection from the GRAPHICS DEVICE.

   (MAIN MENU IS PRESENTED)
```

3.3.2 Make a Chart File

This option provides the user a way to create new charts. This option may be used to make a chart file by doing the following: (1) place the chart or map on the locator pad in a secure manner; (2) select MAKE A CHART FILE from the chart menu; (3) enter the filename to be used

for the chart; and, (4) trace the chart or map as a polyline using the locator. Note: any images appearing on the graphics screen before MAKE A CHART FILE is selected will NOT be stored as part of the chart. Only those images drawn using the MAKE A CHART FILE option will be stored in the file. (This is discussed in greater detail below.)

When MAKE A CHART FILE is selected the user will be prompted to ENTER THE FILE NAME. The user may enter any name for the file. The QUIT MENU (figure A-3) will then appear and the user will be prompted to ENTER A CHART POINT OR QUIT.

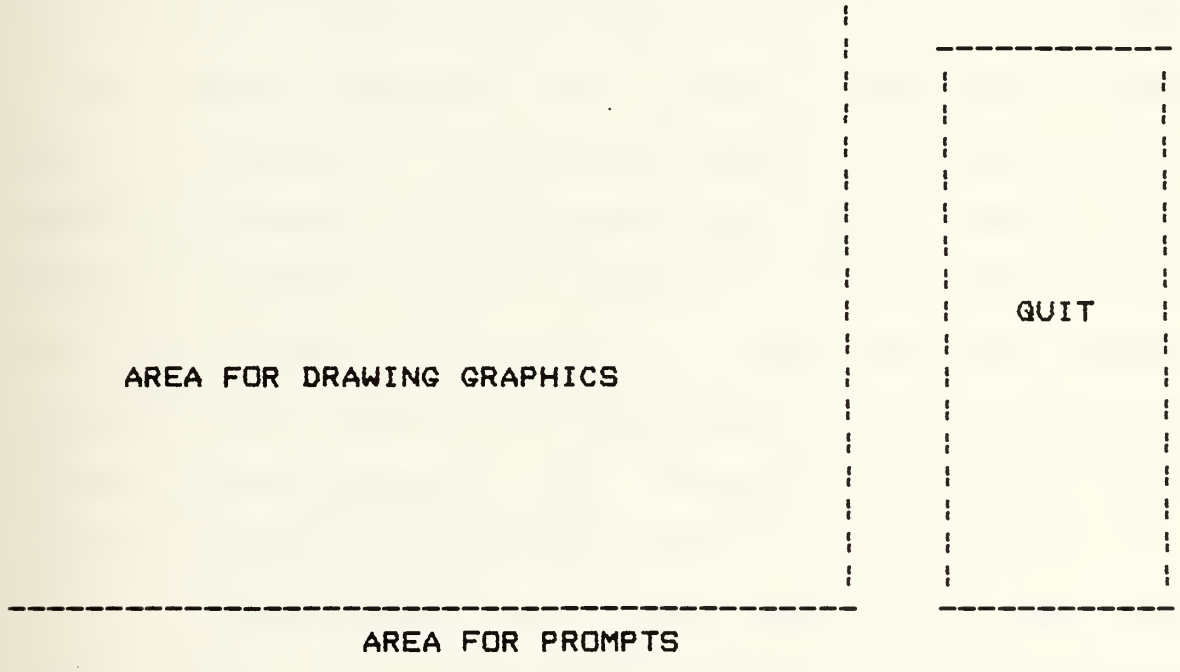

Figure A-3. Quit Menu

The user must then mark the first point of the chart by moving the locator to that position and depressing AND releasing one of the buttons on the locator. A small,

temporary "O" will appear at the location marked. The user may then continue to enter the remaining points of the chart by repeatedly relocating the cursor and depressing AND releasing one of the buttons on the locator, (ie: position the locator, depress AND release a button on the locator, reposition the locator, etc.)

The chart will be drawn as the points are entered and the current point number (first, second, third, etc) and its coordinate values will be echoed on the VT100/102 terminal screen. The chart may have any number of points. Although transparent to the user, each 100 chart points uses one segment, of the 1000 that may be visible at any one time on the graphics terminal, (ie: a map of 1000 points would use 10 segments). A warning message will appear at the VT100/102 terminal, and a warning bell will ring at the VT100/102 terminal, when less than five of the 1000 segments are left. To move the locator to a new position without drawing a line, the user must move the locator with a button on the locator depressed, (ie: depress a button on the locator, reposition the locator, release the button).

When finished tracing the chart, the user must select QUIT. The chart file will be created, the sketch menu will reappear, and the user will be prompted to make another selection from the menu.

Charts may be of any shape and size. A point of the chart may be marked outside the drawing area, but the chart will only be drawn to the edge of the drawing area. Charts are drawn using the same procedure as that employed to draw polylines (refer to LINES below). Charts are drawn using the white color attribute and cannot contain "filled" segments (ie: the fill attribute cannot be "filled"). Although charts may be deleted, copied, and moved while using Chart and Sketch, it is NOT advised as charts which are made up of multiple segments will have to be deleted, copied, or moved segment by segment.

3.3.3 Examples

In order to make the program readily demonstratable and useable, example charts are provided and may be drawn by selecting their name on the CHART MENU, (ie: INDIA, South AMERica, and the MEDditerranean).

3.3.4 Help

The user can receive detailed instructions for using the chart menu, specifically the MAKE A CHART FILE option, by selecting HELP. The instructions are displayed on the VT100/102 terminal.

3.3.5 Quit

If QUIT is selected the user will be returned to the main menu.

3.4 SKETCH MENU

When SKETCH is selected from the main menu, the user is presented with the SKETCH menu and the prompt SELECT FROM MENU. Selections from this menu identify the type (rectangle, diamond, circle, polygon, line, arc, text), color (red, green, yellow, blue, magneta, cyan, white, none), and fill (filled or hollow) characteristics of the graphics segment to be created. It also allows previously created segments to be copied or deleted.

The color and fill attributes of the segments created by the user will be the same as the color and fill attributes of the segments displayed in the menu. (ie: If the menu is written in CYAN, the active color characteristic is CYAN. If the rectangle, diamond, circle, and polygon are hollow, the active FILL characteristic is HOLLOW. Any segment created will be cyan and hollow.)

The SKETCH menu is always presented using the current fill and color attributes. This precludes the user having to remember the current fill or color. When the QUIT option is selected from the SKETCH menu, the user is returned to the MAIN menu. The user may alternate between the MAIN menu and the SKETCH menu as often as desired.

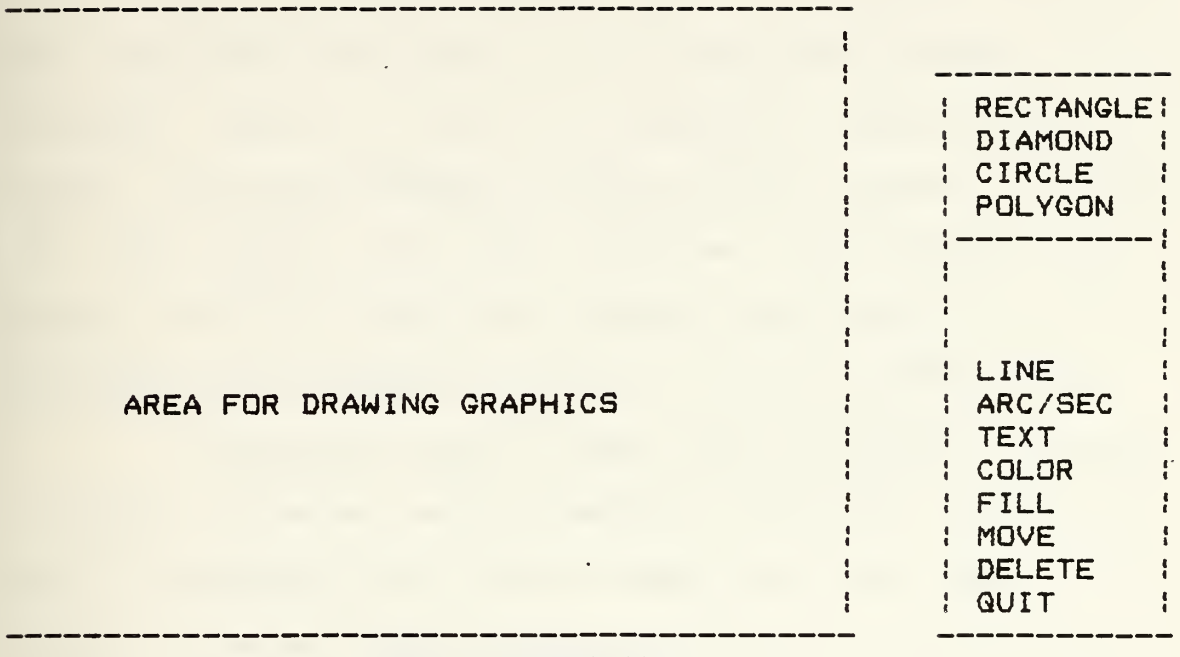

Figure A-4. Sketch Menu

3.4.1 Rectangles

Selecting RECTANGLE (appearing in the menu as a filled or hollow square) allows the creation of horizontally or vertically oriented rectangles. When RECTANGLE is selected, the menu will disappear and the prompt, MARK A CORNER, will be presented. The user must then mark one corner of the rectangle to be drawn by moving the cursor to that location and depressing and releasing one of the buttons on the locator. A small, temporary "O" will appear at the location marked. (The "O" is provided as feedback to the user, markng the location entered on the screen. It will disappear when the rectangle is drawn.)

After the corner has been marked, the prompt MARK THE DIAGONALLY OPPOSITE CORNER, will be presented. The user must again move the cursor, this time to the corner of the rectangle which is to be diagonally opposite the first corner, and again depress a button on the locator. A second "O" will appear at this location, the rectangle will be drawn, the sketch menu will reappear, and the user will be prompted to make another selection from the SKETCH menu.

Rectangles may be drawn with any height and width. A corner location may be marked outside the drawing area, but the rectangle will only be drawn to the edge of the drawing area and the user will not see a "O" confirming the location of that selected corner.

3.4.2 Diamonds

Selecting DIAMONDS allows the creation of diamonds with points oriented at the cardinal headings. When DIAMOND (appearing in the menu as a hollow or filled diamond) is selected, the menu will disappear and the prompt MARK THE CENTER will be presented. The user must then mark the center of the diamond to be drawn by moving the cursor to that location and depressing and releasing one of the buttons on the locator. A small, temporary "O" will appear at the location marked.

After marking the center, the prompt MARK THE RADIUS will be presented. The diamond is drawn as a four-sided circle. The radius is the distance from the

center of the diamond to its points. The user must again move the cursor, this time to the radius of the four-sided circle (diamond) and again depress a button on the locator. A second "O" will appear at this location, the diamond will be drawn, the sketch menu will reappear, and the user will be prompted to make another selection from the menu.

Diamonds may be drawn with any center and radius. A center and/or radius may be marked outside the drawing area, but the diamond will only be drawn to the edge of the drawing area and the user will not see a "O" confirming the location of a center or radius marked outside the drawing area.

3.4.3 Circles

If CIRCLE (appearing in the menu as a hollow or filled circle) is selected the menu will disappear and the user will receive the prompt, MARK THE CENTER. The user must then mark the center of the circle to be drawn by moving the cursor to that location and depressing and releasing one of the buttons on the locator. A small, temporary "O" will appear at the location marked.

The user will then receive the prompt, MARK THE RADIUS. The user must then mark any point which will be on the radius when the circle is drawn by moving the cursor to that location and depressing and releasing a button on the locator. A second "O" will appear at this location, the circle will be drawn, the sketch menu will reappear, and the

user will be prompted to make another selection from the SKETCH menu options.

Circles may be drawn with any center and radius. A center and/or radius may be marked outside the drawing area, but the circle will only be drawn to the edge of the drawing area and the user will not see a "O" confirming the location of a center or radius marked outside the drawing area.

3.4.4 Polygons

If POLYGON (appearing in the menu as a hollow or filled hexagon) is selected the QUIT menu (refer to figure A-3) will appear and the user will be prompted to ENTER A POINT OR QUIT.

The user must then mark the first point of the polygon to be drawn by moving the cursor to that location and depressing AND releasing one of the buttons on the locator. A small, temporary "O" will appear at the location marked.

The user may then continue to enter the remaining points of the polygon by repeatedly relocating the cursor and depressing AND releasing one of the buttons on the locator, (ie: position the cursor, depress AND release a button on the locator, reposition the cursor, etc.) The polygon will be drawn as the points are entered and the current point number (first, second, third, etc) and its coordinate values will be echoed on the VT100/102 terminal

screen. POLYGONS MAY NOT HAVE MORE THAN 100 POINTS. A warning message and bell will be presented at the VT100/102 terminal when less than five of the total 100 points are left.

The user may complete the polygon by moving the cursor back to the original point of the polygon, but it is not neccessary. An implicit line will be drawn from the last point entered to the first when QUIT is selected or when the user moves the cursor to the first point of a new polygon.

To begin a new polygon, without quitting and reselecting polygon, the user must move the locator (and hence the cursor) with a button on the locator depressed, (ie: depress a button on the locator, reposition the locator, release the button). The previous polygon will be closed with an implicit line from the last point to the first.

When QUIT is selected, the last polygon will be closed and the interior filled using the current color and fill attributes (refer to FILL below), the sketch menu will reappear, and the user will be prompted to make another selection from the menu.

Polygons may be any shape and size. A point may be marked outside the drawing area, but the polygon will only be drawn to the edge of the drawing area. POLYGONS MAY NOT HAVE MORE THAN 100 POINTS.

3.4.5 Lines

If LINE is selected the QUIT menu (refer to figure A-3) will appear and the user will be prompted to ENTER A POINT OR QUIT. The user must then mark the first point of the polyline to be drawn by moving the cursor to that location and depressing AND releasing one of the buttons on the locator. A small, temporary "O" will appear at the location marked.

The user may then continue to enter the remaining points of the polyline by repeatedly relocating the cursor and depressing AND releasing one of the buttons on the locator, (ie: position the cursor, depress AND release a button on the locator, reposition the cursor, etc.) The polyline will be drawn as the points are entered and the current point number (first, second, third, etc) and its coordinate values will be echoed on the VT100/102 terminal screen. The polyline may have any number of points, but polylines of more than 100 points will be represented internally (to the program) as multiple segments. The impact of multiple segments is realized when attempting to copy, move, or delete a graphic image. If the image is made up of more than one segment, each segment of the image must be individually copied, moved, or deleted.

The user may complete the polyline by selecting QUIT or by moving the cursor to the first point of a new polyline.

To begin a new polyline, the user must move the locator (and hence the cursor) with a button on the locator depressed, (ie: depress a button on the locator, reposition the locator, release the button). No line will be drawn from the old polyline to the new one.

When QUIT is selected the sketch menu will reappear, and the user will be prompted to make another selection from the menu.

Polylines may be any shape and size. A point of the polyline may be marked outside the drawing area, but the polyline will only be drawn to the edge of the drawing area. Although LINE may be used to draw a closed polygon, it cannot be filled. Select POLYGON to draw a filled polygon. The following table provides a quick reference to the differences between the features available when selecting chart, polygon, or line.

SELECTION	FILL	COLORS
Chart	hollow	white
Polygon	hollow/filled	all
Line	hollow	all

Table 1. Feature Table

3.4.6 Arcs and Sectors

If ARC/SEC is selected the menu will disappear and the user will receive the prompt, MARK THE CENTER. The user must then mark the center of an imaginary circle from which the arc is to be segmented by moving the cursor to that location and depressing and releasing one of the buttons on

the locator. A small, temporary "O" will appear at the location marked.

The user will receive the prompt MARK THE STARTING ANGLE. The user must then mark the end of a line which begins at the center point and which represents the angle FROM which the arc will be drawn. After marking this point, a temporary line will be drawn indicating the starting angle. The Arc will be drawn counter-clockwise from the starting angle.

Another prompt will then be displayed: MARK THE ENDING ANGLE. The user must then mark the end of a line which begins at the center point and which represents the angle TO which the arc will be drawn. After marking this point, a temporary line will be drawn indicating the ending angle.

The user will then receive the prompt, MARK THE RADIUS. The user must then mark any point which will be on the arc when it is drawn by moving the cursor and depressing and releasing a button on the locator. The temporary lines will disappear, the arc will be drawn, the sketch menu will reappear, and the user will be prompted to make another selection from the sketch menu.

If an arc is drawn with the fill characteristic equal to "fill", a filled SECTOR rather than an arc will be drawn. Arcs may be drawn using any center, radius, starting, or ending angles. The center, radius, starting

and ending angles may be marked outside the drawing area, but the arc will only be drawn to the edge of the drawing area and the user will not see a "O" confirming the location of a center, radius, or other point marked outside the drawing area.

3.4.7 Text

If TEXT is selected the menu will disappear and the prompt MARK THE POSITION OF LEFTMOST CHARACTER will appear. The user must then mark the location where the text string should begin by moving the cursor to that location and depressing and releasing one of the buttons on the locator. A small, temporary "O" will appear at the location marked and the TEXT menu will appear.

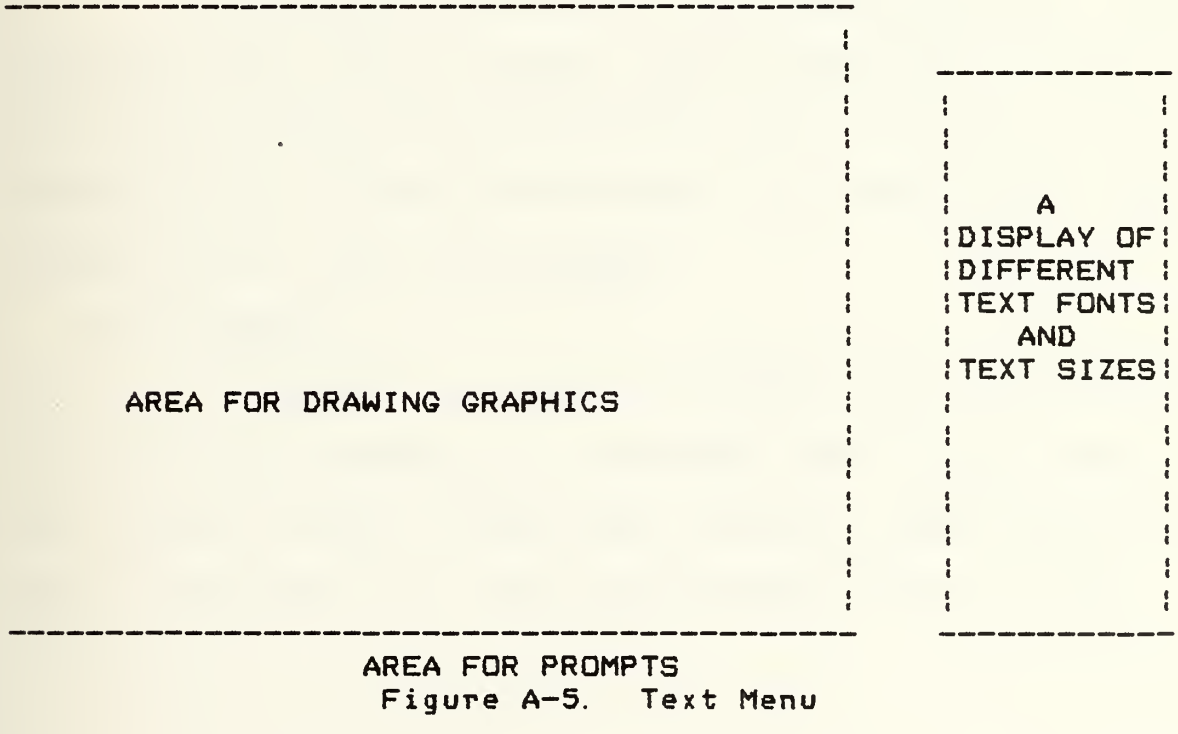

Figure A-5. Text Menu

The user will receive the prompt SELECT FROM MENU. The user must then select one of the examples of text from the menu, to indicate the size and font style desired, by moving the cursor over the desired menu item and depressing and releasing one of the locator buttons. After a selection is made, the TEXT menu will disappear and the user will be promted to enter the text string from the VT100/102 keyboard.

After entering the text string (followed by a carriage return) , the user will be promted to return to the graphics device, the text string will be written on the graphics monitor, the SKETCH menu will reappear and the user will be promted to select from the menu once more.

Text may be written from any starting point. The starting point may be marked outside the drawing area, but the text will only be written within the boundary of the drawing area. If the text string is too long to fit in the drawing area it will be truncated.

3.4.8 Colors

If COLOR is selected the SKETCH menu will will be replaced by the COLOR menu. The COLOR menu displays each of the available colors. Each color option is drawn using the color it decribes, (ie: the option RED is drawn in red).

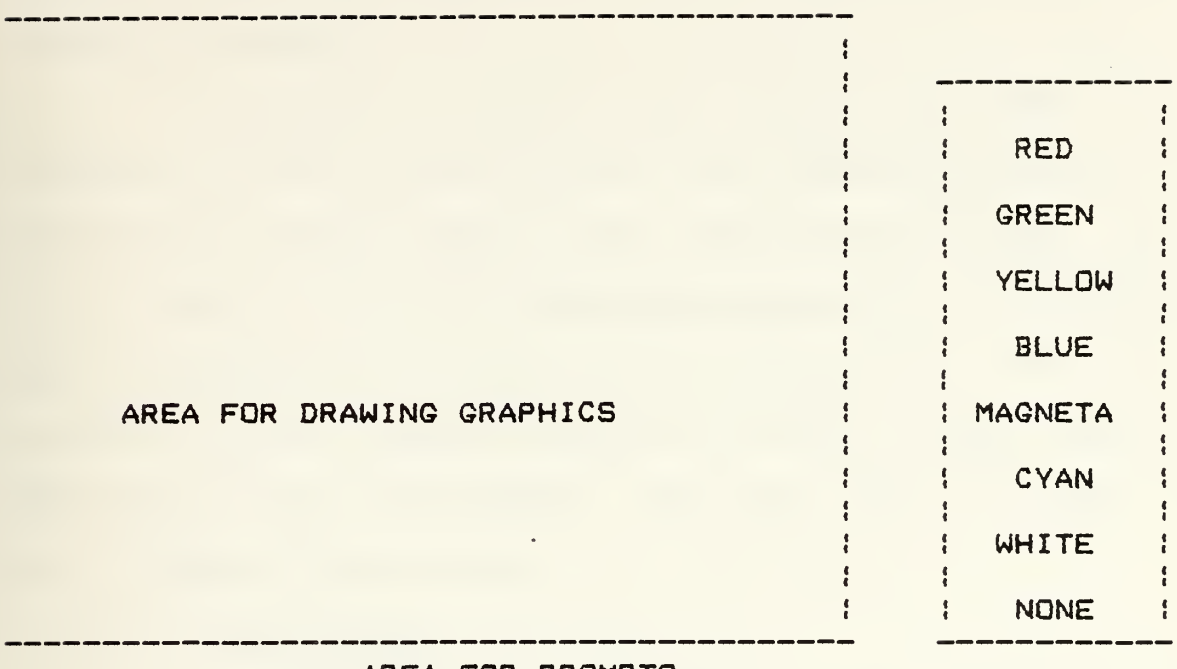

Figure A-6. Color Menu

The user will receive the prompt SELECT FROM MENU. The user must then select one of the colors from the menu by moving the cursor over the desired color and depressing and releasing one of the locator buttons. After a selection is made, the COLOR menu will disappear and the the SKETCH menu will reappear, but written in the color just selected. The SKETCH menu is always displayed using the currently selected color attribute.

All subsequently created graphics will be drawn in the selected color. The color attribute may be changed any number of times before a segment is drawn, but cannot be changed after the type of graphics to be drawn (rectangle, diamond, circle, polygon, line, arc, text) has been selected. Once a segment has been drawn its color cannot be

changed. The color attribute must be selected before the segment is created.

If graphics segments are drawn such that they intersect on the graphics monitor, the RAMTEK RM-9460 may or may not mix the two colors at the intersecting points. This is a feature of the DI-3000 implementation of the RAMTEK that is unavoidable in the current configuration. The only exception to this feature occurs if the one of the colors is NONE (which looks like WHITE). Any color can be drawn on top of NONE without mixing.

3.4.9 Fill Option

The FILL option on the SKETCH menu controls the fill attribute to be used in drawing graphics segments. Selecting the FILL option toggles the fill attribute between HOLLOW and FILLED.

The currently selected fill attribute is the same as that used to display the sketch menu. If the rectangle, diamond, circle, and polygon are hollow, the current fill attribute is HOLLOW. If the rectangle, diamond, circle, and polygon are filled, the current fill attribute is FILLED.

If the user selects FILL, the SKETCH menu will be erased and redisplayed using the new FILL attribute. The SKETCH menu is always displayed using the current COLOR and FILL attributes.

All subsequently created graphics will be drawn using the current fill attribute. The fill attribute may be changed any number of times before a segment is drawn, but cannot be changed after the type of graphics to be drawn (rectangle, diamond, circle, polygon, line, arc, text) has been selected. Once a segment has been drawn the fill attribute cannot be changed.

The DI-3000 implementation of the RAMTEK RM-9460 fills segments starting from the center. If a "filled" graphics segment is drawn on top of another segment, the latter segment will only be "filled" in the none-intersecting region that contains the center of the graphics segment. If there are overlapping colors, they may or may not be mixed. These are features of the DI-3000 implementation of the RAMTEK hardware that is unavoidable in its current configuration.

3.4.10 Deletions

DELETE provides the user with the ability to selectivly delete (erase) any graphics segment drawn using SKETCH. If DELETE is selected, the SKETCH menu will be replaced by the UPDATE menu and the prompt, MARK DELETIONS will appear, (refer to figure A-7).

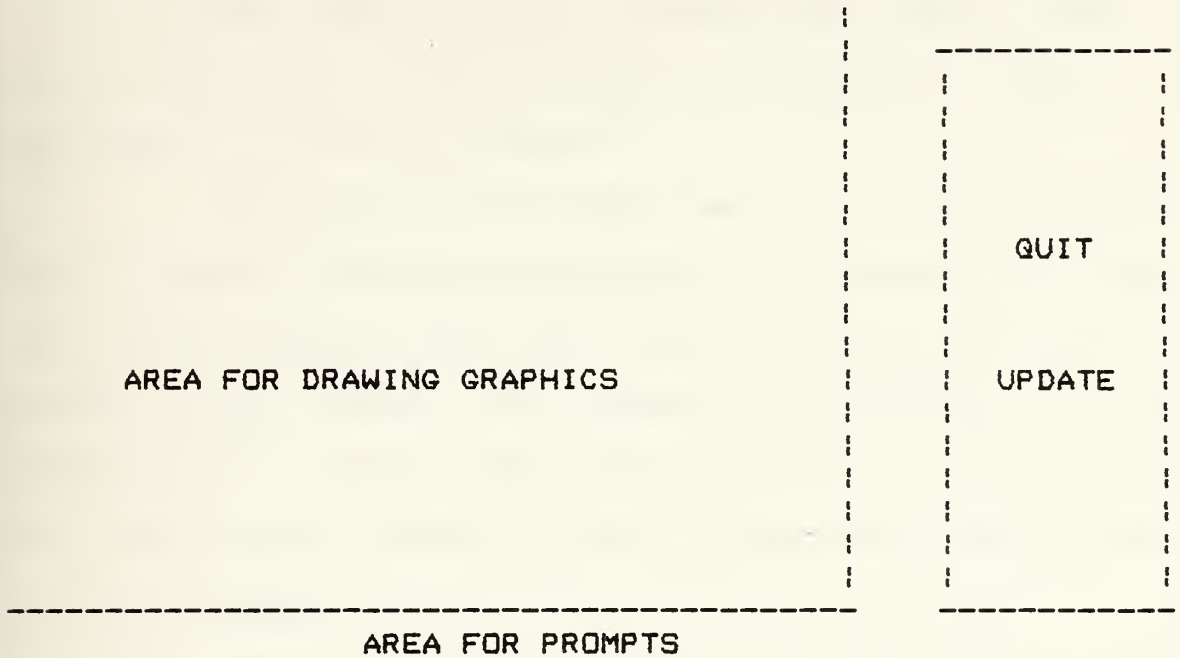

Figure A-7. Update Menu

The user must then mark some part of the segment to be deleted by moving the cursor to that location and depressing and releasing one of the buttons on the locator. After a segment has been marked it is immediately deleted. The DI-3000 does not always do a good job of this. The DI-3000 does not always erase the deleted segment completely. The user can force a redrawing of the graphics (to complete the erasure) by selecting UPDATE from the menu. An update is automatic when the user selects QUIT.

If the user depresses a button on the locator and the cursor is not over a graphics segment, no action will occur, and the prompt, MARK DELETIONS will continue to be displayed.

The user may make as many deletions as desired. If the user marks a location outside the drawing area, no action will occur. Once a segment is deleted it cannot be UNdeleted. It must be redrawn.

To return to the SKETCH menu the user must select QUIT from the UPDATE menu (figure A-7) by moving the cursor over the QUIT option and and depressing and releasing a button on the locator. All segments not completely deleted before will be deleted, the sketch menu will reappear, and the user will be prompted to make a selection from the menu.

3.4.11 Copies

Chart and Sketch provides the facility to make copies of graphics segments which have already been drawn using Sketch. If COPY is selected, the UPDATE menu menu (figure A-7) will appear and the prompt, MARK THE SEGMENT will be presented. The user must then mark some part of the segment to be copied by moving the cursor to that location and depressing and releasing one of the buttons on the locator.

After a segment has been marked, the prompt, MARK THE NEW LOCATION will be presented. The user must move the cursor to the location where the copy is to be drawn, and again depress a button on the locator. A copy of the marked segment will be drawn at the new location, using the currently selected color attribute, and the user will be prompted to mark another segment or select QUIT.

The DI-3000 does not always erase the old (moved) segment completely. The user can force a redrawing of the graphics (to complete the erasure) by selecting UPDATE from the menu. An update is automatic when the user selects QUIT.

If the user depresses a button on the locator and the cursor is not over a graphics segment, no action will occur, and the prompt, MARK THE SEGMENT will continue to be displayed.

The user may make as many copies of a segment as desired at any location(s). If the user selects a new location outside the drawing area, only that portion of the segment which would be within the drawing area will be visible. Copies are made using the original FILL attribute. It is possible to make copies of copies. When the user selects QUIT the sketch menu will reappear, and the user will be prompted to make another selection from the sketch menu.

3.4.12 Quitting the Sketch Menu

The QUIT option is used to return to the MAIN menu. No data is lost in returning to the MAIN menu. If QUIT is accidently selected from the SKETCH menu, the user can return to exactly the same place in the program by reselecting SKETCH from the MAIN menu.

3.5 CREATING FILES

The FILE option on the MAIN menu provides the user with a way to save the graphics that have been created using Chart and Sketch by encoding the picture currently displayed on the graphics device and writing it to a file in the user's directory.

After selecting FILE from the main menu the user will be prompted to enter the name of the file into which the encoded data should be written. If the user specifies an already existing file, a new version be made. If the user specifies a file which does not exist, it will be created. The user will also be prompted to enter the filetype to be used, either a SECTION file or a DATA file.

A "data" file stores the logical constructs on the screen. It uses minimal disk space and can be modified after being stored and recalled. Data files take longer to redisplay and cannot be used by any program (other than Chart and Sketch) to recreate the filed graphics. It is not interprogram portable. However, due to the small amount of storage used, this method of storage should be used if at all possible.

A "section" file stores the current display on the on the screen, pixel by pixel. Since this method provides rapid redisplay of graphics it might be best used for briefings. The section file is created and read using a DI-3000 graphics routine and is therefore interprogram

portable. (Any program written in DI-3000 graphics language can read a section file created by Chart and Sketch.) However, section files use large amounts of storage and cannot be modified after they are created.

A chart, created using the CREATE A CHART FILE option from the CHART menu, can only be retrieved by selecting READ A CHART FILE from the CHART menu. (Charts have a filetype of MAP). The following table summerizes the differences between data, map, and section files as implemented in Chart and Sketch.

FILE TYPE	DATA	MAP	SECTION
Retrieval speed	slow	slow	fast
Storage size	small	small	large
Changeable ?	yes	yes	no
Portable ?	no	no	yes
Colors	all	white	all
Retrieve using:	main menu	chart menu	main menu
Filetype	.DAT	.MAP	.SEC

Table 2. Chart and Sketch Filetypes

If the user selects to use a SECTION file the current picture on the graphics device will be redrawn to fill the entire screen (without menus or prompts) and the file will be created.

An error in writing the file will result in an error message. Examples are shown below. The program prompts are indented.

Example 4. Creating a Section File - no errors

Enter the file name.

```
DI3000

  A SECTION OR DATA FILE ? (enter S or D or ?)

?

  A DATA file stores the logical constructs on the
  screen.  It uses minimal disk  space and can be
  modified after being stored and recalled.
  USE THIS METHOD IF YOU CAN.

  A SECTION file stores the contents of the SCREEN.
  It is better for briefings and rapid display, but
  uses large amounts of storage and cannot be
  changed after it is filed.

  Do you want to use a SECTION or DATA file?
  (enter S or D)

S

  (THE CURRENT DISPLAY IS REDRAWN TO FILL THE SCREEN)

  Writing graphics to file.  Please stand by.

  (THE FILE IS CREATED)

  The file    DI3000.SEC      has been written.

  (THE CURRENT DISPLAY IS RESTORED TO ITS ORIGINAL SIZE)

  Enter selection from the GRAPHICS DEVICE.

  (THE MAIN MENU IS DISPLAYED)
```

Example 5. Creating a Section File - with errors

```
  Enter the file name.

DI3000

  A SECTION or DATA file (enter S or D or ?)

S

  (THE CURRENT DISPLAY IS REDRAWN TO FILL THE SCREEN)

  ERROR in creating the file     DI3000.SEC.
```

(THE CURRENT DISPLAY IS RESTORED TO ITS ORIGINAL SIZE)

Enter selection from the GRAPHICS DEVICE.

(THE MAIN MENU IS DISPLAYED)

Example 6. Creating a Data File - no errors

Enter the file name.

DI3000

A SECTION OR DATA FILE ? (enter S or D or ?)

D

Writing graphics to file. Please stand by.

(THE FILE IS CREATED)

The file DI3000.DAT has been written.

Enter selection from the GRAPHICS DEVICE.

(THE MAIN MENU IS DISPLAYED)

The DI-3000 software also returns error messages in some instances which may seem rather cryptic. In most cases this results from insufficient storage space to write the file.

Any number of displays may be FILED by Chart and Sketch, however, section files should be used with care as they require large amounts of storage. Graphics which have been filed may be redisplayed by selecting RETRIEVE from the main menu. (see subsection 3.5)

3.6 RETRIEVING FILES

The RETRIEVE option on the MAIN menu provides the user with a way to display graphics that were previously created and FILED using Chart and Sketch. (see subsection 3.6) The program will prompt the user to enter the filename of the file to be displayed on the graphics device. The user must enter the filename. The user will then be asked the file type, either S for section or D for data. Table 2, Chart and Sketch Filetypes details the differences between data, map, and section files.

If the file is a data file, the user will be asked if the entire drawing area should be utilized or if the graphics should be displayed on only a selected portion of it. The program will prompt the user (on the VT100/102) to enter the filename of the file to be displayed on the graphics device. The user must enter the filename (from the VT100/102). At this time two things will happen, (1) the verify menu will appear on the graphics screen with the prompt, USE FULL VIEWSPACE ?; and, (2) an explanation of the prompt will appear on the VT100/102 terminal. The user must select YES, to use the full viewing space, or NO, to use only a selected portion of it.

If NO is selected, the user will be prompted, at the graphics terminal, to enter the desired display area as a rectangle. The user will be prompted to enter two diagonally opposite corners of the rectangle. The vertical

aspect of the specified rectangle will be adjusted such that the rectangle in which the graphics are drawn has the same horizontal/vertical ratio as the drawing area, to prevent distortion of the graphics. The graphics will be drawn within the adjusted rectangle. If YES is selected from the verify menu, the graphics will be drawn using the full drawing area. After the graphics have been drawn, the user is returned to the main menu and prompted to make another selection. (See the example below.)

If the file was a section file, the user will be asked if he wants to view another file. If the user enters Y, meaning yes, he will again be asked to enter the filename. If the user enters N, for no, the currently displayed file will be erased and the user will be returned to the MAIN menu and prompted to make a selection.

If an error is encountered in finding or reading the file, the user will be given a short error message and returned to the main MENU with the prompt to select from the menu. An example is shown below. The program prompts are indented, the user responses are not.

Example 7. Retrieving a Section File - no errors

```
  Enter file name.
DI3000
  Is this a SECTION or DATA file ?  (enter S or D or ?)
?
```

```
  If the file you are recalling has filetype SEC
  enter S.  If it has filetype DAT enter D.

  Is this a SECTION or DATA file?
  (PLEASE ENTER S or D)

S

 (FILE IS DISPLAYED)
  Read another file? (Y or N)

Y

  Enter file name.

HOUSE

  (FILE IS DIPLAYED)

  Read another file?

N

  Enter selection from the GRAPHICS DEVICE.
  (SCREEN IS ERASED AND MAIN MENU IS PRESENTED)
```

Example 8. Retrieving a Data File - no errors

```
    Enter the file name.

DI3000

    Is this a SECTION or DATA file ?  (enter S or D or ?)

?

    If the file you are recalling has filetype SEC
    enter S.  If it has filetype DAT enter D.

    Is this a SECTION or DATA file?
    (PLEASE ENTER S or D)

D

    When recalling a data file the user has
    the option of redrawing the graphics to
    their original size (which will use the
    FULL VIEWing SPACE), or to a reduced
```

```
    size (which will use only a selected
    portion of the viewing space.

    DO YOU WANT TO USE THE FULL VIEWSPACE?
    (Enter YES or NO from the graphics terminal.)

NO  --- from the graphics terminal

    (THE USER IS PROMPTED TO ENTER TWO POINTS
    DEFINING A RECTANGLE USING THE LOCATOR)

--- the user enters two points at the graphics terminal

    (THE FILE IS DISPLAYED WITHIN THE RECTANGLE
    AND THE MAIN MENU IS PRESENTED)
```

Example 9. Retrieving a data file - with errors

```
    Enter file name.

DI3010    --- suppose that DI3010 does not exist ---

    Is this a SECTION or DATA file ?  (enter S or D or ?)

?

    If the file you are recalling has filetype SEC
    enter S.  If it has filetype DAT enter D.

    Is this a SECTION or DATA file?
    (PLEASE ENTER S or D)

D

    ERROR in reading file     DI3010.dat .
    Enter selection from the GRAPHICS DEVICE.
    (MAIN MENU IS PRESENTED)
```

Any number of files can be retrieved. Retrieved SECTION files cannot be overlayed (drawn one on top of another) or modified. Retrieved DATA files can be both overlayed and modified.

3.7 CLEARING THE SCREEN

The CLEAR option should only be selected when the user is finished using the current graphics and all graphics that are to be saved have been written to a file using FILE. Once the CLEAR is selected all graphics not saved in a file are lost.

In order to insure that CLEAR was not selected incidentally, the program will require verification of the user's intention to CLEAR. The MAIN menu will disappear, the VERIFY menu with 'YES' and 'NO' options will appear, and the user will be asked, DO YOU REALLY WANT TO CLEAR?

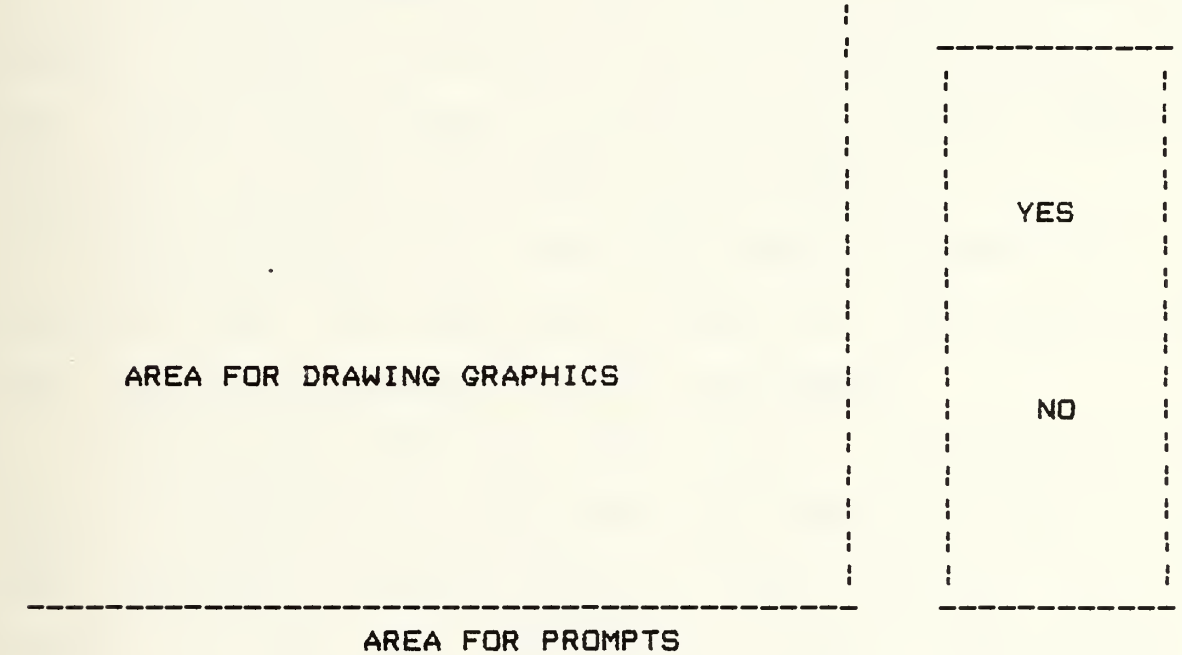

Figure A-8. Verify Menu

If NO is selected, the user will be returned to the MAIN menu and no graphics will be lost. If YES is selected, all graphics will be deleted.

3.8 VIEWING THE MAP

It is often desirable to display the graphics and map without the menus and prompts that usually appear on the graphics device. The VIEW option provides this facility.

When the user selects VIEW, the graphics in the drawing area will be redrawn so as to fill the entire graphics screen. To return to the normal view, the user will be prompted (on the VT100/102) to depress and release one of the buttons on the locator device.

3.9 MOVING SEGMENTS

Any graphics segment in the drawing area can be moved to another location by selecting MOVE from the MAIN menu. If the drawing area is empty (ie: there are no visible graphics segments), selecting the option has no effect.

Assuming there is at least one graphics segment in the drawing area, the UPDATE menu (figure A-7) will appear and the user will be prompted to MARK THE SEGMENT.

The user must then mark some part of the segment to be moved by moving the cursor over that segment and depressing and releasing one of the buttons on the locator. If the cursor is over the intersection of two or more segments, that segment which was most recently created or moved will be selected. If the user depresses a button, and the cursor is not over a graphics segment, no action will occur and MARK THE SEGMENT will continue to be displayed.

After a segment has been marked the user will be prompted to MARK THE NEW LOCATION. The user must move the cursor to the new desired location and depress and release a button on the locator. The segment will be erased and redrawn at the new location using the same fill and color attributes. The DI-3000 does not always erase the old segment completely. The user can force a redrawing of the graphics (to complete the erasure) by selecting UPDATE from the menu. An update is automatic when the user selects QUIT.

A segment may be moved as often as desired. If the user marks a location outside the drawing area, the marked segment will disappear but will not be redrawn. When the user selects QUIT, the graphics will be redrawn, the main menu will reappear and the user will be prompted to make selection from the menu.

3.10 ZOOMING AND PANNING

The ZOOM option provides the user the ability to magnify any part of the drawing area by a power of 2. The graphics screen is too small to display the entire, magnified drawing area at once, so the user may change the center of the view (panning). After selecting the ZOOM option, the user will be prompted (at the VT100/102 monitor), to MOVE THE LOCATOR WITH THE BUTTON DEPRESSED TO PAN. RELEASE THE BUTTON WHEN THE DESIRED VIEW IS ON THE

SCREEN. The user must then depress, and maintain depressed, one of the buttons on the locator. The view on the graphics monitor will be magnified.

By maintaining the button depressed and moving the locator across the face of the graphics tablet, the user can cause the view on the graphics monitor to pan. When the user releases the button on the locator the panning will stop and the picture on the graphics monitor will freeze.

The user will receive the prompt PRESS A BUTTON ON THE LOCATOR TO RETURN TO NORMAL VIEW (again at the VT100/102). The display on the graphics device will remain unchanged until the user depresses and releases one of the locator buttons. When a button is depressed and released the graphics view will return to normal, the MAIN menu will be presented, and the user will be prompted to select from the menu.

3.11 QUITTING

The QUIT option should only be selected when the user is finished using the program and all graphics that are to be saved have been written to a file using FILE. Once the program ends all graphics not saved in a file are lost.

In order to insure that QUIT was not selected accidentally, the program will require the user to verify the intention to QUIT. The MAIN menu will disappear, the VERIFY menu (refer to figure A-8) with 'YES' and 'NO'

options will appear, and the user will be asked, DO YOU REALLY WANT TO QUIT?.

If NO is selected, the user will be returned to the MAIN menu and no graphics will be lost, if YES is selected, the program will end.

4. ADDITIONAL INFORMATION

More detailed information concerning both the software and hardware involved in Chart and Sketch is provided in an M.S. Thesis by James Jay Tschudy: Chart and Sketch: An Interactive Color Graphics Program, Naval Post Graduate School, Monterey, California, 1984. A detailed description of the Precision Visual DI-3000 software used in developing Chart and Sketch can be located in the DI-3000 User's Guide (Precision Visuals, Order No. DI3817, Boulder, Colorado, 1982.

Another M.S. thesis, by Ron Elmlinger (A Tutorial for DI-3000 Programming, Naval Post Graduate School, Monterey, California, 1984) details and examples of the DI-3000 programming language. The War Lab also has an online graphics help function which provides detailed information on the CSL implementation of the DI-3000. This help function can be invoked by the system command $HELP DI3000.

APPENDIX B

DOCUMENTATION OF CSC ROUTINES

1. INTRODUCTION. 124

2. DOCUMENTATION. 125

2.1 ESCAPE CODE 9405, BLOCK PIXEL READ. 125

2.2 ESCAPE 9406, PIXEL WRITE 127

2.3 ESCAPE 9412, HARDWARE ZOOM 128

2.4 ESCAPE 9420, SET VIDEO ORIGIN 128

2.5 ESCAPE 9424, WRITE PIXEL IMAGE FROM SECTION FILE . . . 129

2.6 ESCAPE 9425, TRANSFER PIXEL IMAGE TO SECTION FILE. . . 130

2.7 ESCAPE 9428, READ PIXEL IMAGE INTO MEMORY. 131

2.8 ESCAPE 9423, WRITE PIXEL IMAGE FROM MEMORY. 132

2.9 KSCRET. 133

2.10 KSBYTE24. 134

2.11 KTECHO. 134

2.12 JPICK. 135

1. INTRODUCTION

This appendix documents the graphics routines (referred to in the foregoing thesis) added to the DI-3000 software by the Conflict Simulation Center (CSC). It does not document ALL the routines used in Chart and Sketch. It consists of comments and excerpts from the Development DI-3000 package created by CSC of the Lawrence Livermoore National Laboratory.

This appendix is intended to assist the reader in fully understanding the PROGRAM IMPLEMENTATION section of the foregoing thesis. Documentation of all the routines used in Chart and Sketch can be found in Precision Visuals, DI-3000 User's Guide, (Order No. DI3817), Precision Visuals, Boulder, Colorado, 1982; or, the DI-3000 Development Package Conflict Simulation Laboratory, Lawrence Livermoore National Laboratory, Sunnyvale, CA, 1983.

2. DOCUMENTATION

CSC has made several enhancements to Precision Visuals, Inc. (PVI) graphics package, DI-3000. Most of these enhancements are available as escape codes, others are available as extensions to DI-3000 and others are modifications to the DI-3000 package.

Escape codes provide a means to implement device specific graphics functions without totally sacrificing a graphic standard (DI-3000 uses the SIGGRAPH CORE standard as a guidline). Since the Ramtek instructions are not CORE compatable, DI-3000 could not fully utilize most of the Ramtek instruction set. By implementing escape codes, CSC was able to use the DI-3000 device-independent graphics package and, in critical applications, use escape codes for optimal use of the graphics hardware.

2.1 ESCAPE 9405, BLOCK PIXEL READ.

Read a block of data from a Ramtek 9400 refresh memory into a disk file. As with all pixel image escape codes, at most 8 bit-planes are supported.

CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)

CODE - (integer*4) the escape code number, 9405 for this function.

```
NINTEG -(integer*4) number of integer arguments,
                    1 to 2 for this function.

NREAL - (integer*4) number of real arguments,
                    0 or 4 for this function.

ILIST - (integer*4) array of NINTEG integer arguments.
        ILIST(1) =  Fortran logical unit number assigned to
                    the disk file into which the pixel data
                    is written.
        ILIST(2) =  read mask (optional).  Default is all
                    planes.  The default is used if the
                    value is zero.

RLIST - (real) array of NREAL real arguments:
        RLIST(1),RLIST(2) = lower left corner of window in
                            virtual coordinates.
        RLIST(3),RLIST(4) = upper right corner of window in
                            virtual coordinates.
```

If the window defined by the lower-left corner and upper-right corner is outside of the viewport, an error will result. The first record written to the disk file includes information on the window size.

For example, to read the pixels with all bit planes in the window with the lower-left corner of (-1.0,-0.8) and the upper-right corner of (0.6,0.8) into disk file "pixelout.dat":

```
REAL RARG(4)
.
.
.
RARG(1) = -1.0
RARG(2) = -0.8
RARG(3) = 0.6
RARG(4) = 0.8
IS = LIB$GET_LUN(LUN)
IF (.NOT.IS) CALL LIB$SIGNAL(%VAL(IS))

OPEN (UNIT=LUN, NAME='PIXELOUT.DAT', TYPE='NEW',
+       ACCESS='SEQUENTIAL', FORM='UNFORMATTED',
+       ORGANIZATION='SEQUENTIAL', BLOCKSIZE=16384,
+       RECORDSIZE=4088, INITIALSIZE=2588)
```

```
Call block pixel write escape function.
CALL JESCAP (9405, 1, 4, LUN, RARG)

Close file.
CLOSE (UNIT=LUN)
CALL LIB$FREE_LUN(LUN)
```

2.2 ESCAPE 9406, PIXEL WRITE

Write a block of pixel data from a disk file into the Ramtek 9400 refresh memory. As with all pixel image escape codes, at most 8 bit-planes are supported.

```
CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)
```

CODE - (integer*4) the escape code number, 9406 for this function.

NINTEG -(integer*4) number of integer arguments, 1 to 2 for this function.

NREAL - (integer*4) number of real arguments, 0 or 2 for this function.

ILIST - (integer*4) array of NINTEG integer arguments:
- ILIST(1) = Fortran logical unit number assigned to the disk file from which the pixel data is read.
- ILIST(2) = write mask (optional). Default is all planes. The default value is used if the mask value is zero.

RLIST - (real) array of NREAL real arguments:
- RLIST(1),= lower left corner of window in
- RLIST(2) virtual coordinates. This window size will remain the same as when the file was written.

2.3 ESCAPE 9412, HARDWARE ZOOM

This is of limited usefulness since all views on any MCP are affected by the firmware zoom. With Ramtek 9460 controllers, however, DI-3000 should be configured so each view (workstation) has a different MCP. The video origin may be set (default is (0,0)) using escape code 9420. Also see echo level 10 for the LOCATOR (this is demonstrated in program DEMOS:LOC).

CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)

CODE - (integer*4) the escape code number, 9412 for this function.

NINTEG -(integer*4) number of integer arguments, 1 for this function.

NREAL - (integer*4) number of real arguments, 0 for this function.

ILIST - (integer*4) zoom factor via pixel replication, 0 through 15. No zoom is a zoom value of 0.

RLIST - dummy for this function.

Example: zoom from the lower left corner of the screen by a factor of 4 and then return to no zoom.

```
...
CALL JESCAP(9412,1,0,3, )
...
CALL JESCAP(9412,1,0,0, )
...
```

2.4 ESCAPE 9420, SET VIDEO ORIGIN

With this escape code the video origin may be set to determine the window viewed after a zoom. With a zoom of greater than one the video origin may be continually updated

to perform a pan over the entire, original window. Also, LOCATOR echo 10 may be used to perform the panning locally, in the Ramtek controller (see program DEMOS:LOC).

CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)

CODE - (integer*4) the escape code number, 9420 for this function.

NINTEG -(integer*4) number of integer arguments, 0 for this function.

NREAL - (integer*4) number of real arguments, 2 for this function.

ILIST - dummy

RLIST - (real) array of NREAL real arguments.
RLIST(1) = X (screen) coordinate of new video origin.
RLIST(2) = Y (screen) coordinate of new video origin.

2.5 ESCAPE 9424, WRITE PIXEL IMAGE FROM SECTION FILE

Write a properly formatted section file to all connected Ramteks. This is the fastest mechanism to put pixel image data from a file to the Ramtek screen. To generate the properly formatted section files use program DEMOS:CONSEC or escape code 9429. As with all pixel image escape codes, at most 8 bit-planes are supported.

CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)

CODE - (integer*4) the escape code number, 9424 for this function.

NINTEG -(integer*4) number of integer arguments, 1 or 2.

NREAL - (integer*4) number of real arguments, 0 or 2.

ILIST - (integer*4) list of arguments:
ILIST(1) = first address of the section file as returned by KSOPEN.

ILIST(2) = (optional) bit plane write mask.

RLIST - list of real arguments if any. The 2 elements are the x- and y-coordinate of the lower-left corner of the window into which the pixels are written. If not given, the same window used to read the pixel (from escape code 9405) are used.

2.6 ESCAPE 9425, TRANSFER PIXEL IMAGE TO SECTION FILE

Read a window of pixels from the Ramtek and write a properly formatted section file. The section file must already have been created and mapped into the users' virtual address space using KSCRET (to create and map section file) or KSOPEN (to open and map section file). As with all pixel image escape codes, at most 8 bit-planes are supported.

CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)

CODE - (integer*4) the escape code number, 9425 for this function.

NINTEG -(integer*4) number of integer arguments, 1 or 2.

NREAL - (integer*4) number of real arguments, 0 or 4.

ILIST - (integer*4) list of integer arguments:
ILIST(1) = first address of the section file as returned by KSOPEN/KSCRET.
ILIST(2) = (optional) bit plane read mask.

RLIST - (real) list of real arguments if any. The 2 elements are the x- and y-coordinate of the lower-left corner of the window into which the pixels are written. If not given, the same window used to read the pixel (from escape code 9405) are used.

2.7 ESCAPE 9428, READ PIXEL IMAGE INTO MEMORY

Read a block of pixels (one byte per pixel) into an array. As with all pixel image escape codes, at most 8 bit-planes are supported.

CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)

CODE - (integer*4) the escape code number, 9428 for this function.

NINTEG -(integer*4) number of integer arguments, 1 or 2.

NREAL - (integer*4) number of real arguments, 4.

ILIST - (integer*4) list of arguments:
ILIST(1) = address of buffer of at least the size needed to hold all bytes (one byte per pixel) in the window.
ILIST(2) = (optional) plane read mask. Defaults to all planes ('FF'X).

RLIST - (real) the lower-left corner and upper-right corner (in virtual coordinates) of the window to read.

The returned buffer is the value of the pixels in the given window. The pixels start at the lower-left corner and procede left to right, bottom to top. There is one byte per pixel.

The array address MUST be on a byte boundary (safest and fastest if INTEGER*4 boundary). Use routine KSBYTE to get the size of the array in bytes. Or use KSLENX and KSLENY to get the dimensions of the array and calculate the area. Use escape 9428 and 9429 together for reading/writing pixel windows. It is compatable with neither 9405/6 nor 9424/25.

Example: read the pixels from window [(0.0,0.0),(0.5,0.6)] and set the background in this window to plane 1 only. Use a dynamically allocated array. The statements for a declared are commented out.

```
INTEGER*4 BIGARRAY(250000)

RARG(1) = 0.0
RARG(2) = 0.0
RARG(3) = 0.5
RARG(4) = 0.6

IARG(1) = %LOC(BIGARRAY)

NBYTES = KSBYTE(RARG(1),RARG(2),RARG(3),RARG(4))
IX     = KSLENX(1,RARG(1),RARG(3))
IY     = KSLENY(1,RARG(2),RARG(4))
NBYTES = IX*IY
IS     = LIB$GET_VM(NBYTES, IARG(1))
IF (.NOT.IS) CALL LIB$STOP(%VAL(IS))
CALL JESCAP(9428, 1, 4, IARG, RARG)
CALL BYTEOP(%VAL(IARG(1)), NBYTES)
IARG(2) = 1
CALL JESCAP(9429, 2, 4, IARG, RARG)
.
.
SUBROUTINE BYTEOP(IBUF, LEN)
BYTE IBUF(LEN)
DO I=1,LEN
  IF (IBUF(I).EQ.0) IBUF(I) = 'FF'X
ENDDO
RETURN
END
```

2.8 ESCAPE 9423, WRITE PIXEL IMAGE FROM MEMORY

Write the window of pixels (one byte per pixel) to the Ramtek. As with all pixel image escape codes, at most 8 bit-planes are supported.

```
CALL JESCAP (CODE, NINTEG, NREAL, ILIST, RLIST)

CODE -   (integer*4) the escape code number, 9429 for
                     this function.
```

NINTEG - (integer*4) number of integer arguments, 1 or 2.

NREAL - (integer*4) number of real arguments, 4.

ILIST - (integer*4) list of arguments:

ILIST(1) = address of array of pixel values.
ILIST(2) = (optional) bitplane write mask.
Defaults to 'FF'x (all planes).

RLIST - (real) window specified by the lower-left corner and the upper-right corner in virtual coordinates.

2.9 KSCRET

A section file is created, of the specified size, and mapped into the users virtual address space. From then on the section file is accessable as program data.

CALL KSCRET(FILENAME, ISIZE, LUN, ICHANNEL, IADDR, IERR)

FILENAME = (character) name of the section file to be created, a character string.

ISIZE = (integer*4) size of the file in bytes. The created file will be rouned up to an integral number of blocks (512 bytes).

LUN = (integer*4) logical unit number on which the section is opened is returned.

ICHANNEL = (integer*4) channel number on which the section file is opened. This channel must be deassigned (using SYS$DASSGN) after the section's virtual address space is deallocated to disassociate the section file from the process.

IADDR = (integer*4) array of 2 elements at which the starting and ending address of the section file now exists.

IERR = (integer*4) error returned if the file could not be opened. In addition, KSCRET will display a message in that case. If an error occurs while mapping the section file the program will be terminated with the system error message displayed.

2.10 KSBYTE24

This routine will return the number of bytes necessary to read the pixels in the given window. Overhead is included for the escape codes 9424/9425 only for KSBYTE24.

CALL KSBYTE(DSPDEV, VXLL, VYLL, VXUR, VYUR)

CALL KSBYTE24(DSPDEV, VXLL, VYLL, VXUR, VYUR)

DSPDEV = (integer*4) display device number on which the window is to be measured. This device must be ENABLED and ON (JDINIT and JDEVON).

VXLL,VYLL = (real) lower-left corner of window, in virtual coordinates.

VXUR,VYUR = (real) upper-right corner of window, in virtual coordinates.

2.11 KTECHO

Routine KTECHO is available to assist in fully using all echo level functions. The call is as follows:

CALL KTECHO(DSPDEV, INPFCT, PHYDEV, NI, NR, IARG, RARG)

DSPDEV - (integer*4) display device.

INPFCT - (integer*4) input type (=2 for locator).

PHYDEV - (integer*4) physical device

(1 or 2 for locator).

NI - (integer*4) number of integer arguments.

NR - (integer*4) number of real arguments = 2* number of coordinates.

IARG - (integer*4) array of NI integer arguments.

RARG - (real) array of coordinates in the order X1, Y1, X2, Y2, . . .

For echo level 10:

NI = 2, NR = 4

IARG(1) = echo level(=10).
IARG(2) = zoom factor (0-15).

RARG(1),RARG(2) = virtual lower-left corner of window to be panned.

RARG(3),RARG(4) = virtual upper-right corner of window.

Echo level 10 is a "pan". The screen is mapped into the corresponding screen window indicated by the real arguments. This mapping is used to put the video origin at the cursor position. Hence panning will be performed only within the window given. If the zoom is not great enough, the panning may not fill the entire screen.

2.12 JPICK

The PICK function implies a LOCATE function. If the LOCATE returns a positive button value, the position of the puck is used to perform a PICK. Please see the documentation on JLOCAT for details.

CALL JPICK(DSPDEV, PHYDEV, ECHOLV, BUTTON, SEGNAM, PICKID)

DSPDEV - (integer*4) selected display device.

PHYDEV - (integer*4) physical device of the PICK device.

ECHOLV - (integer*4) echo level. Only echo level 7 is supported, all other echo levels result in echo level 1 (the screen cursor tracks the graphics tablet puck).

BUTTON - (integer*4) button value from the PICK operation. It is the bit map of the button(s) depressed on the puck at the time of the PICK, it is negative

if the pick failed or it is zero if no buttons have just been depressed.

SEGNAM - (integer*4) segment number picked, less than zero if none picked.

PICKID - (integer*4) pickid with the segment picked, less than zero if none picked.

JPICK will pick with the location performed only when the button is depressed.

APPENDIX C

HARDWARE AND SOFTWARE REQUIREMENTS

TABLE OF CONTENTS

1. HARDWARE REQUIREMENTS. 138

2. SOFTWARE REQUIREMENTS. 140

3. PHYSICAL REQUIREMENTS. 142

1. HARDWARE REQUIREMENTS

Chart and Sketch was developed on a Digital Equipment Corporation VAX 11/780. The operating system used is VMS. (VAX stands for Virtual Address Extension and VMS stands for Virtual Memory System.) The FORTRAN portion of Chart and Sketch can be run on any system that supports FORTRAN-77 and has some type of system executive. The graphics commands are generated by calling upon the Precision Visual DI-3000 integrated software graphics routines. The DI-3000, being a two-level package, accepts device-independent graphics calls at the user level. These calls are interpreted by the device driver at the graphics display device level. The device driver is an interface within the DI-3000 to the specific display device, the RAMTEK RM-9460. The device driver translates the device-independent graphics commands into the RAMTEK RM-9460 device-dependent commands. [Ref. 1]

The program also utilizes additional graphics escape routines provided by the CSC (Conflict Simulation Center, Lawrence Livermoore, CA) implementation of DI-3000. For the program to run properly, the host must be supported by the CSC implementation of the DI-3000.

The program requires a host computer alphanumeric terminal. Any type of alphanumeric terminal may be used to run Chart and Sketch. There are no commands unique to a specific type of alphanumeric terminal. The only terminal I/O commands are accept, type, read, and write. A hardcopy terminal may be used, but the clear_screen procedure may waste paper as it prints 30 blank lines.

The program also requires at least one graphics display device, associated device driver, and locator/pick input devices. DI-3000 is a device independent graphics tool. Chart and Sketch can be run on many differing graphics systems if provided the appropriate device driver.

2. SOFTWARE REQUIREMENTS

The VAX/VMS operating system [Ref. 9 and 10] must be used to run Chart and Sketch as written. The software packages required are the VAX/VMS file executive and VAX-11 FORTRAN (FORTRAN-77) [Ref. 11 and 12], and very few system services.

Conversion to another computer system with a different operating system, but with some dialect of FORTRAN-77, would require only minor adjustments in most of the program (ie, input and output requirements such as open, close, type and accept). Two subroutines, Retrieve_sec and File_sec would require substantial testing and modification. Most systems have some form of file executive, however, the retrieve and file subroutines would have to be transposed line by line into the new system command language and the CSC graphic routines used in Retrieve_sec and File_sec would have to be tested and modified because of their dependence on the VAX/VMS operating system.

Conversion to another computer system with a similar operating system that supports only FORTRAN-4 would require some modifications in the program. FORTRAN-4 does not include the 'IF...THEN...ELSE' construct, TYPE or ACCEPT statments, or character type variables. In conversion to

FORTRAN-4, the extensively used 'IF-THEN-ELSE' contructs in Chart and Sketch would have to be replaced with multiple 'IF' and 'GOTO' statements. The TYPE and ACCEPT statements would have to be converted to PRINT and READ statements with the appropriate FORMAT statements.

3. PHYSICAL REQUIREMENTS

A User's Manual should be available to the user of the program, however, Chart and Sketch is highly interactive and very user friendly. It is possible that the program could be successfully used without a User's Manual and with very minimal instruction.

Further information about the VAX computer system may be located in one of the following DEC Manuals: the VAX/VMS Primer [Ref. 13], the VAX/VMS Command Language User's Guide [Ref. 14] and the VAX-11 FORTRAN Language Reference Manual [Ref. 15].

LIST OF REFERENCES

1. Precision Visuals, DI-3000 User's Guide, (Order No. DI3817) Precision Visuals, Boulder, Colorado, 1982.

2. Conflict Simulation Center, DI-3000 Development Package, Lawrence Livermoore National Laboratory, Sunnyvale, CA, 1983.

3. Ramtek Corporation, RM-9460 Graphic Display System Hardware Reference Manual, (Order No. 8000080-02, Revision A) RAMTEK Corporation, Santa Clara, California, 1983.

4. Foley, J. D. and Van Dam, A., Fundamentals of Interactive Computer Graphics, pp. 55-59, Addison-Wesley Publishing Co., 1983.

5. Ibid., pp. 206-208.

6. Miller, R. B., "Response Time in Man-Computer Conversational Transactions", AFIPS Conf. Proc., v. 33, pp. 267-277, 1968.

7. Graham, N., Introduction to Computer Science, pp. 73-202, West Publishing Co., 1982.

8. NOSC, Interim Battle Group Tacitical Trainer (IBGTT), System Capabilities Description, NOSC, 1983.

9. Digital Software, VAX/VMS Guide to Using Command Procedures, (Order No. AA-H782B-TE) Digital Equipment Corporation, Maynard, Massachusetts, 1982.

10. Digital Software, VAX-11 System Services Reference Manual, (Order No. AA-D018C-TE) Digital Equipment Corporation, Maynard, Massachusetts, 1982.

11. Digital Software, VAX/VMS Command Language User's Guide, (Order No. AA-D023C-TE) Digital Equipment Corporation, Maynard, Massachusetts, 1982.

12. Digital Software, VAX-11 FORTRAN User's Guide, (Order No. AA-D035C-TE) Digital Equipment Corporation, Maynard, Massachusetts, 1982.

13. Digital Software, VAX/VMS Primer, (Order No. AA-D030B-TE) Digital Equipment Corporation, Maynard, Massachusetts, 1980.

14. Digital Software, VAX/VMS Command Language User's Guide, (Order No. AA-D023C-TE) Digital Equipment Corporation, Maynard, Massachusetts, 1982.

15. Digital Software, VAX-11 FORTRAN Language Reference Manual, (Order No. AA-D034C-TE) Digital Equipment Corporation, Maynard, Massachusetts, 1982.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Professor M. Sovereign, Code 74 Chairman C3 Academic Group Naval Postgraduate School Monterey, California 93943	1
4.	CDR G. Porter, Code 55Pt Director, Gaming War Lab Naval Postgraduate School Monterey, California 93943	4
5.	Capt James Tschudy 926 E. Mathewson Placentia, California 92670	1
6.	Lt Col J. Malokas, Code 39 C3 Curricular Officer Naval Postgraduate School Monterey, California 93943	1
7.	AFIT/CIRD ATTN: Capt B. Van Orman Wright-Patterson AFB, Ohio 45433	1
8.	Naval Ocean Systems Center, Code 8242 ATTN: Dennis Mc Call San Diego, California 92152	1
9.	U.S. Army War College ATTN: Maj Howard Yellen, USA Carlisle Barracks, Pennsylvania 17013	1
10.	Naval War College Center for War Gaming ATTN: CDR R. Adams Newport, Rhode Island 02840	1

11. Dr. A. M. Zied, Code 39 — 1
Technical Director, War Lab
Naval Postgraduate School
Monterey, California 93943

12. Major General D. C. Evans — 1
WIS JPMO
Washington, D.C. 20330

207764

Thesis
T8229 Tschudy
c.1 CHART AND SKETCH: an interactive computer graphics program.

207764

Thesis
T8229 Tschudy
c.1 CHART AND SKETCH: an interactive computer graphics program.

AMERICAN BINDERY TOPEKA KANSAS

thes18229
CHART AND SKETCH :
3 2768 000 98383 7
DUDLEY KNOX LIBRARY